

Entity Reference with Layout may be the Zen of editorial layout control!

Really, more about editorial experience...



About Me

Scott Sawyer - Freelance Developer

Drupal.org 2008

<https://www.drupal.org/u/scottsawyer>

<https://twitter.com/ScottSawyer>

<https://www.scottsawyerconsulting.com>

Slack: Scott Sawyer

Always looking for work :)



Why Bother?

Drupal has a perception problem. Drupal is underrated.

We need to grow the install base.

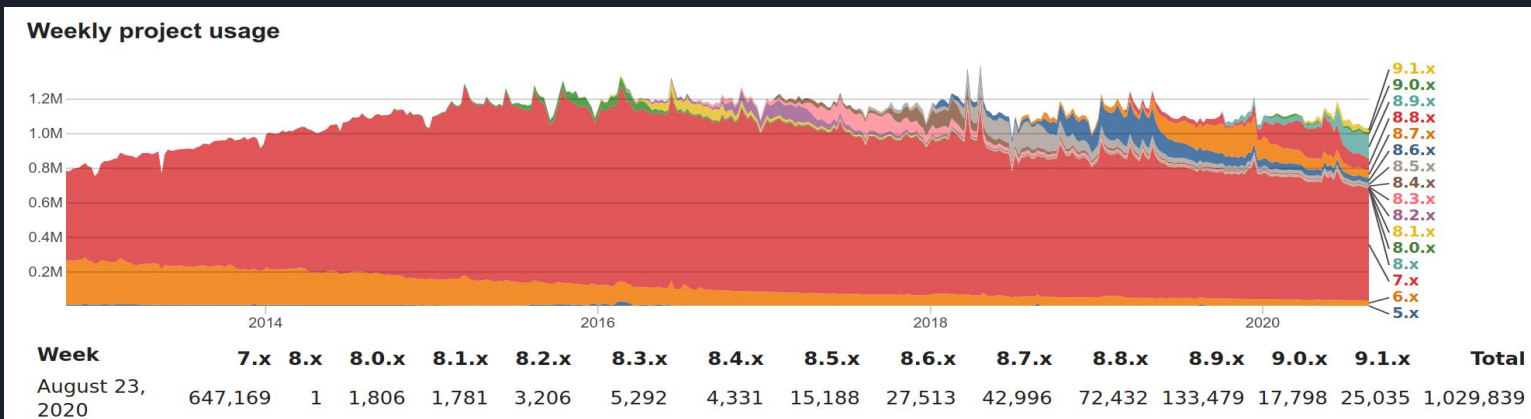
We need more developers making money with Drupal.

Installs

D7: 647,169

D8: 308,024

D9: 42,833





How is Drupal perceived?

Common criticisms

- Drupal is hard.
- Drupal is confusing.
- Drupal is not user friendly.

Reality

- Certainly can be.
- Options vs decisions.
- Let's fix this today.

What is the problem?

Drupal has so many ways to control content display.

- Field Group
- Panels
- Display Suite
- Blocks
- Layout Builder
- Etc.

Each has pros and cons, work in different ways.

None work in the content editor.

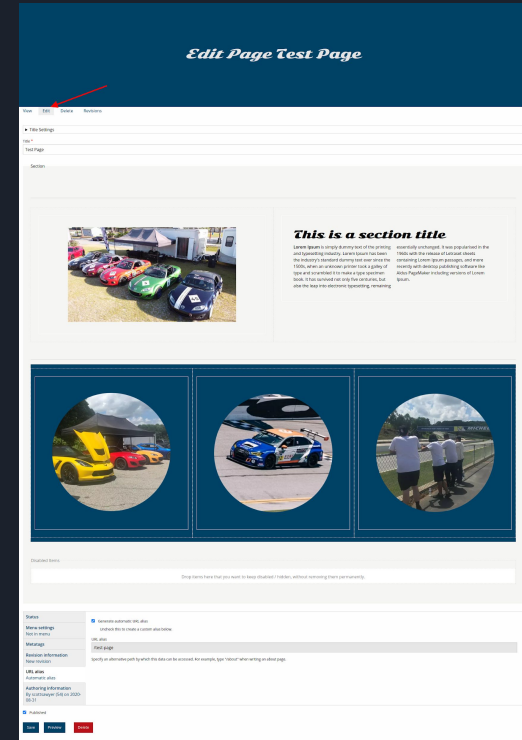
Site Owners / Content Editors just want to build the content, not search through menus.



What's the solution?

As site builders / developers:

- Focus on user tasks.
- Simplify the admin.
- Unify the site editing interfaces.

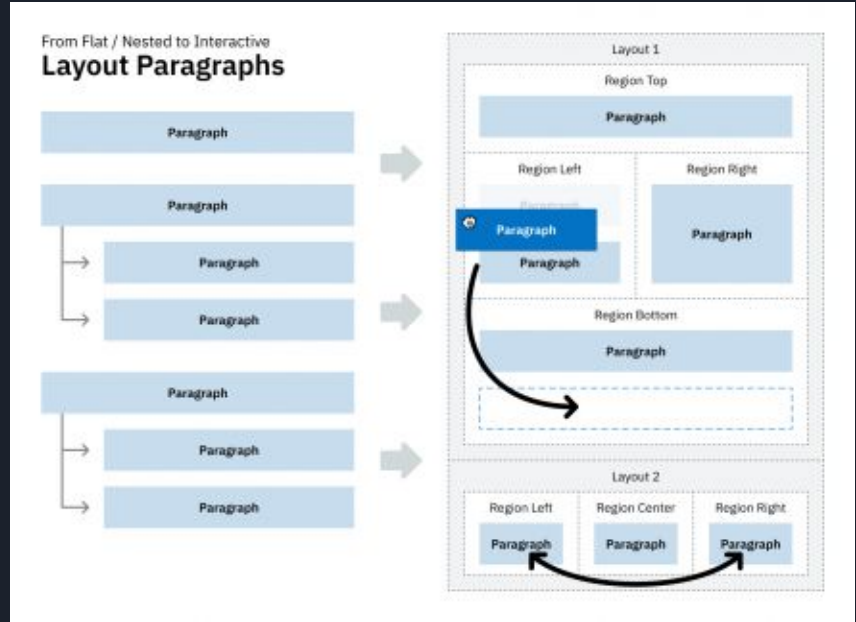


Entity Reference Layout

Power of layouts, simplicity of a single UI.

Uses Paragraphs for creating content.


*NOTE: Entity Reference Layout replaced by Layout Paragraphs.





Let's talk about Layouts


Drupal default Layouts are not great.


Select a layout: *

 One column

 Two column


 Two column bricks


 Three column
25/50/25


 Three column
33/34/33


Custom Layouts with more flexibility.

Select a layout: *

 Four Column

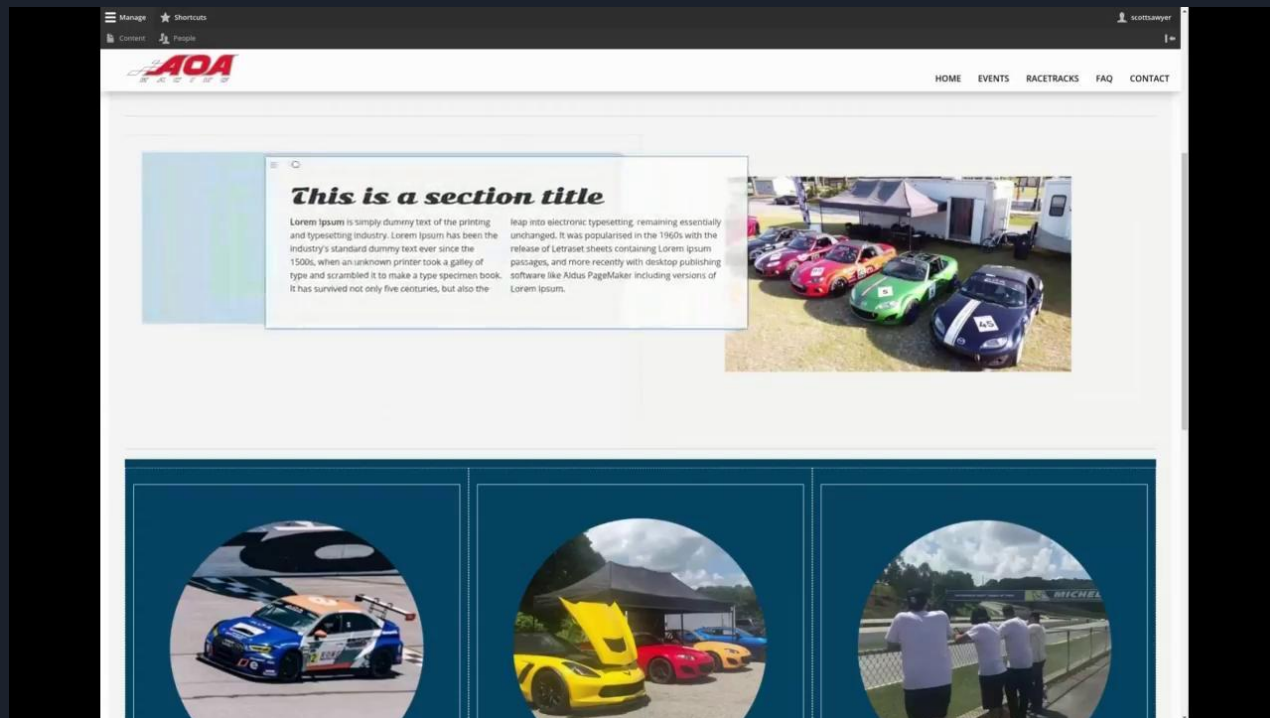
 One Column

 Three Column

 Two Column

Demo

<https://bit.ly/3imGoDo>



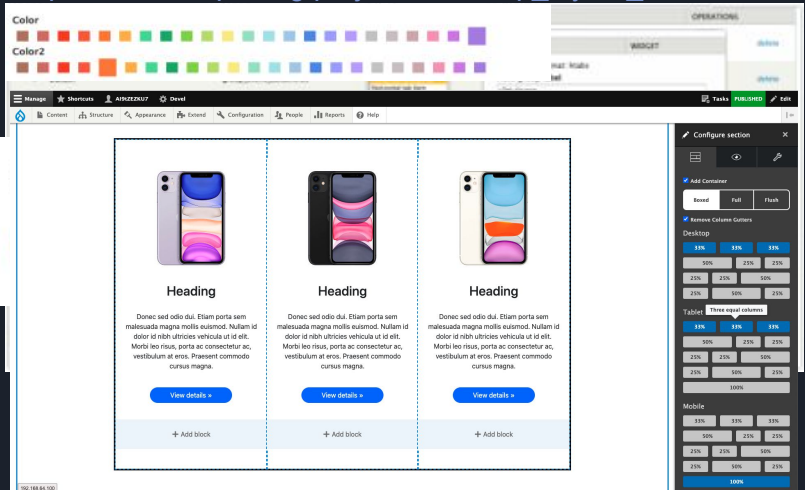
Helper Modules

Modules to help make the editor experience more visual.

Custom layouts or something like Bootstrap
Grid or Dropwidge
Layout Builder.

https://www.drupal.org/project/color_field_widget

https://www.drupal.org/project/bootstrap_layout_builder



General Principles

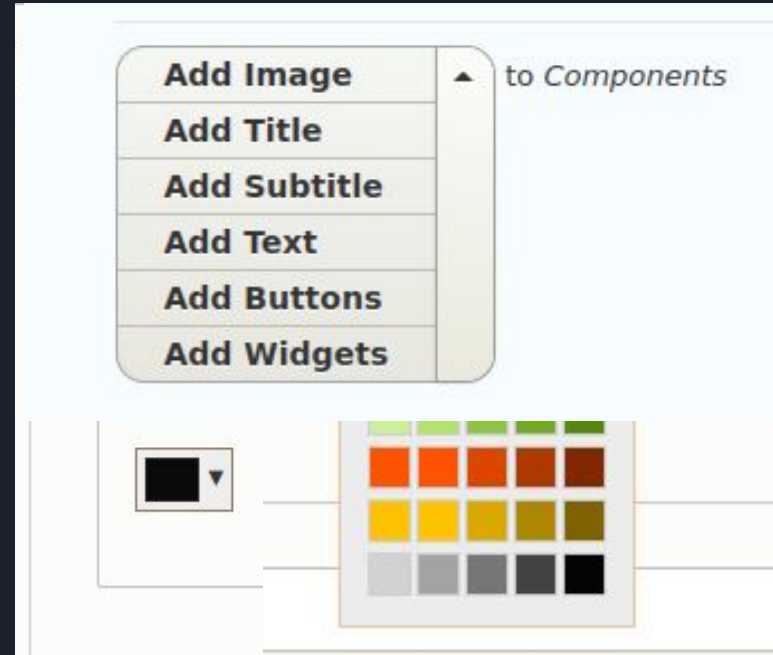
Simplify controls.

Inline help.

Sensible defaults.

Enforce branding.

Leverage Components.



Use Field Group to streamline content editing

Home Administration Structure Paragraphs types Content

Manage form display ☆

Edit Manage fields **Manage form display** Manage display

+ Add field group

Show row weights

Field	Widget	
Settings	Details	Default state closed Mark as required delete
Settings Tabs	Tabs	Direction: horizontal delete
Styling	Tab	Tab: closed delete
Background Color	Color spectrum	
Background Opacity	Range Slider	
Padding	Tab	Tab: closed delete

Settings

Background Size Padding Alignment

Configure section background. NOTE: Configure only one background type.

Background Color

Background Image

Show media item weights

Remove

car-4.png
The maximum number of media items have been selected.

Background Attachment

Fixed

Background Video

Components

[Home](#) > [Administration](#) > [Structure](#)

Paragraphs types ☆

+ Add paragraph type

Icon	Label	Machine name	Description	Operations
	Button	button	A button.	Manage fields ▼
	Buttons	buttons	Button group	Manage fields ▼
	Content	content	A container for content	Manage fields ▼
	Image	image	An image field.	Manage fields ▼
	Remote Video	remote_video	A field for displaying a YouTube or Vimeo field.	Manage fields ▼
	Section	section	A section or stripe container.	Manage fields ▼
	Text	text	A text area for entering content.	Manage fields ▼
	Title	title	A title element	Manage fields ▼
	Video	video	Local Video field.	Manage fields ▼
	Widgets	widgets	Widgets field.	Manage fields ▼

Componentize Code

Define Style Guide

```
1 <?php
2 /**
3  * Define Color Palette.
4  * @return array $colors
5  */
6 function color_palette() {
7     $palette = [
8         'light-blue' => '#33b5ff',
9         'dark-blue' => '#004166',
10        'red' => '#b9121b',
11        'orange' => '#ff5b27',
12        'beige' => '#b6abab',
13        'dark-brown' => '#e44444',
14        'light-gray' => '#e6e6e6',
15        'medium-gray' => '#cacaca',
16        'dark-gray' => '#8a8a8a',
17        'white' => '#fefefe',
18        'black' => '#0a0a0a',
19    ];
20    foreach ($palette as $key => $value) {
21        $parts = preg_split('/-/', $key, -1);
22        $label = ucfirst(implode(' ', $parts));
23        $machine_name = implode('-', $parts);
24        $colors[] = ['label' => $label, 'machine_name' => $machine_name, 'class' => $key, 'code' => $value];
25    }
26    return $colors;
27 }
28 /**
29  * Match color code to machine name.
30  *
31  * @param string $code
32  *
33  * @return string $machine_name || NULL
34  */
35 function color_code_match($code) {
36     if (strpos($code, '#') === FALSE) {
37         $code = '#' . $code;
38     }
39     $color_palette = color_palette();
40     foreach ($color_palette as $index => $props) {
41         if ($code == $props['code']) {
42             return $props['class'];
43         }
44     }
45     return NULL;
46 }
```

Preprocess Paragraphs

```
1 <?php
2 /**
3  * Implements hook_preprocess_paragraph_type().
4  */
5 function THEME_preprocess_paragraph_content(array &$vars) {
6     $p = $vars['paragraph'];
7     $classes[] = 'content--config';
8     // Background color.
9     $background_color_classes = _build_background_color_classes($p);
10    if (!empty($background_color_classes)) {
11        for ($i = 0; $i < count($background_color_classes); $i++) {
12            $classes[] = $background_color_classes[$i];
13        }
14    }
15    // Padding classes.
16    $padding_classes = _build_padding_classes($p);
17    if (!empty($padding_classes)) {
18        foreach ($padding_classes as $class) {
19            $classes[] = $class;
20        }
21    }
22    // Append classes.
23    if (!empty($classes)) {
24        for ($i = 0; $i < count($classes); $i++) {
25            $vars['attributes']['class'][] = $classes[$i];
26        }
27    }
28 }
```

Sass Component

```
1 /** PARAGRAPHS: Content */
2
3 .content--config {
4     @include background-color;
5     @include padding;
6 }
7
8 // ====
9 // Background color
10 // Creates styles from background color classes.
11 // ====
12
13 @mixin background-color ($elm: null) {
14     @each $color, $val in $palette {
15         @each $opacity in $opacities {
16             $o: $opacity * 1;
17             @if $elm == null {
18                 &.background-color--#{ $color }.background-color-opacity--#{ $opacity } {
19                     background-color: rgba($val, $o);
20                 }
21             }
22             @else {
23                 &.background-color--#{ $color }.background-color-opacity--#{ $opacity } #{ $elm } {
24                     background-color: rgba($val, $o);
25                 }
26             }
27         }
28     }
29 }
```



Use Cases

Do use when:

- Editors are not familiar with Drupal.
- When you are willing to spend some time curating options.
- When pages need maximum flexibility.

Don't use when:

- Content Layout needs to be identical.
- When you don't have time to structure backend forms.



Questions?