



DrupalCon
EUROPE2020
DECEMBER 8-11

An Overview of Drupal Front-End Component Integration Methods

Brian Perry
Lead Front-End Developer
bounteous

Slides & Sandbox Repo:
<http://bit.ly/component-int>

BRIAN PERRY

- Lead Front End Dev at Bounteous
- Rocking the Chicago suburbs
- Lover of all things components...
...and Nintendo



d.o: brianperry

twitter: bricomedy

github: backlineint

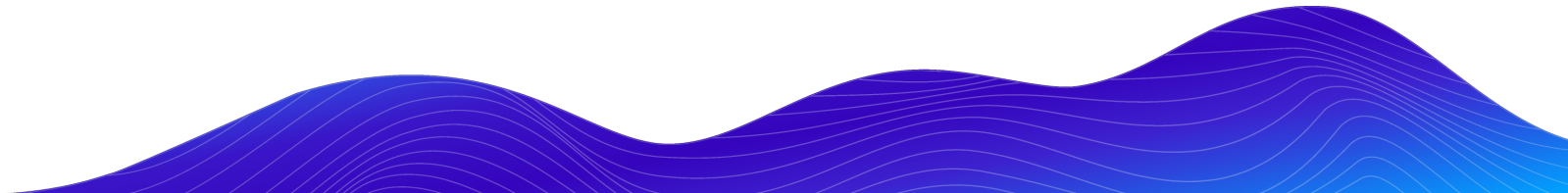
nintendo: wabrian

brianperryinteractive.com

bounteous



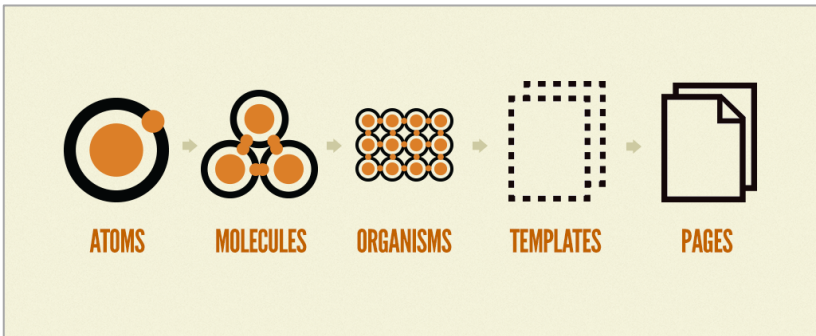
COMPONENTS!



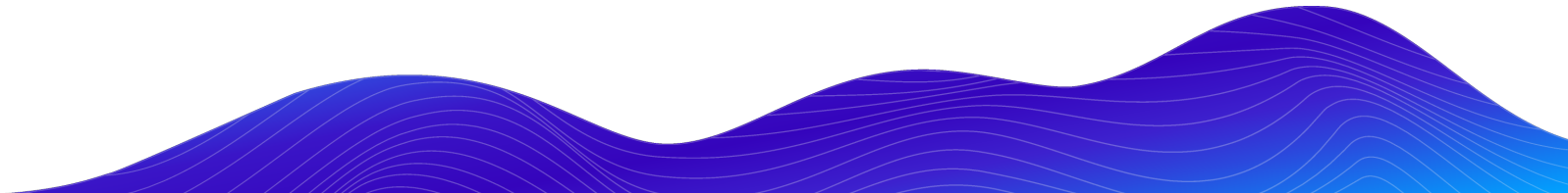
COMPONENT BASED DEVELOPMENT

What is it?

- Creating modular and re-usable elements
- Building a design system, not a series of pages
- Can use a pattern library for documentation and prototyping
 - Tools like Pattern Lab and Storybook
- Can help decouple front and back end development.



OUR EXAMPLE TWIG COMPONENT



Containers

Container.is-centered

Good morning. Thou hast had a good night's sleep, I hope.

Container.is-dark

Good morning. Thou hast had a good night's sleep, I hope.

Good morning. Thou hast had a good night's sleep, I hope.

Good morning. Thou hast had a good night's sleep, I hope.



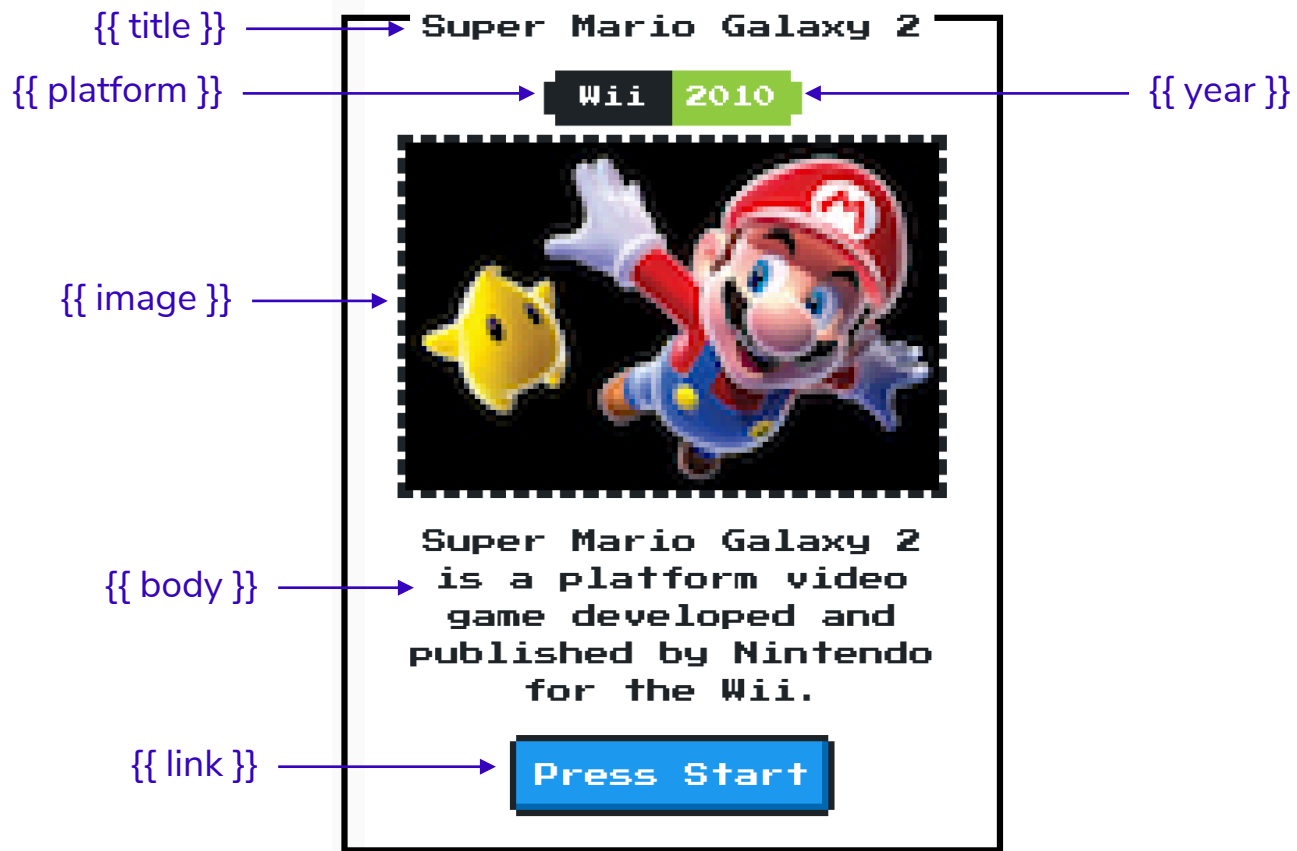
Super Mario Galaxy 2

Wii 2010



Super Mario Galaxy 2 is a platform video game developed and published by Nintendo for the Wii.

Press Start



```

c-container.twig web/themes/custom/nes/nes-components/source/_patterns/03-components/container/c-container
<div class="nes-container with-title is-centered">
  {% if title %}
    <p class="title">{{ title }}</p>
  {% endif %}
  {% if platform or year %}
    <div class="nes-badge is-splited">
      {% if platform %}<span class="is-dark">{{ platform }}</span>{% endif %}
      {% if year %}<span class="is-success">{{ year }}</span>{% endif %}
    </div>
  {% endif %}
  {% if image %}
    {{ image }}
  {% endif %}
  {% if body %}
    {{ body }}
  {% endif %}
  {% if link %}
    <a class="nes-btn is-primary" href={{ link }}>Press Start</a>
  {% endif %}
</div>

```

Super Mario Galaxy 2

Wii 2010



Super Mario Galaxy 2
is a platform video
game developed and
published by Nintendo
for the Wii.

Press Start

```
! c-container.yml web/themes/custom/nes/nes-components/source/_patterns/03-components/c
title: Container
platform: Wii
year: 2010
image: >
  
body: <p>Super Mario Galaxy 2 is a platform video game developed and
link: '#'
```

Super Mario Galaxy 2

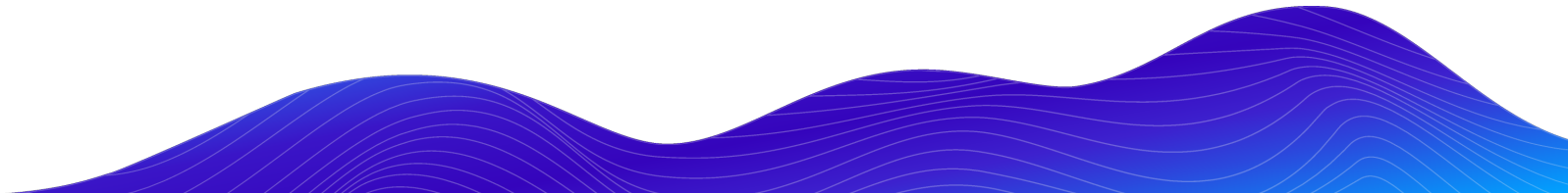
Wii 2010



Super Mario Galaxy 2
is a platform video
game developed and
published by Nintendo
for the Wii.

Press Start

COMPONENTS IN DRUPAL



WHERE DO MY COMPONENTS LIVE?

For the sake of this talk...

Standard Drupal Components

- Live in the default template directory
- May not require any additional effort to get data to display

Integrated Drupal Components

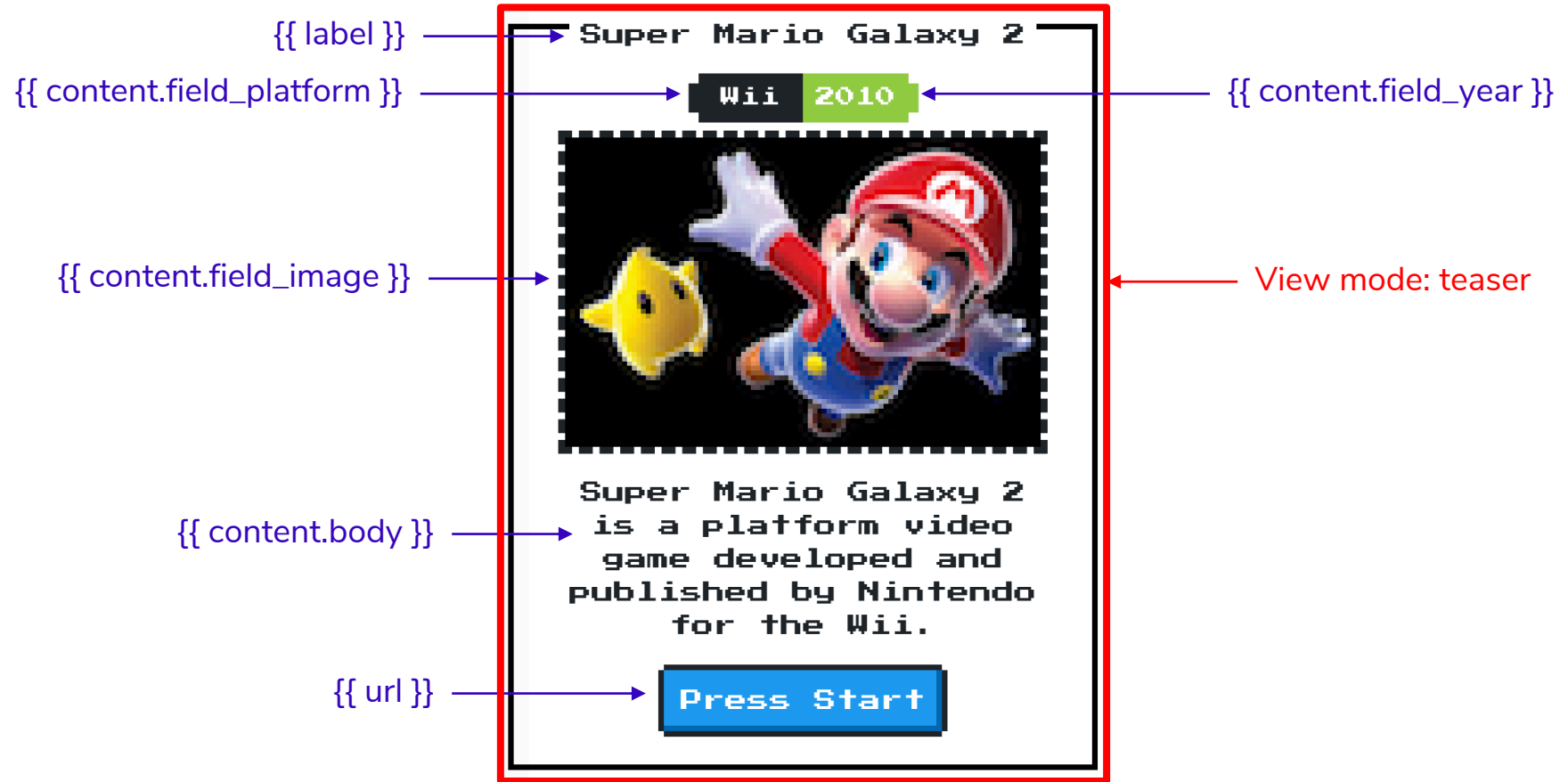
- Live somewhere other than the default templates directory
- Require some additional effort to get data to display
- For this talk, I don't really care how your integrated components get into your theme.
 - Could live in your theme
 - Could be external dependency

STANDARD DRUPAL COMPONENTS

May be right for your team or project. No shame necessary.

- Build with Drupal (and only Drupal) in mind.
- Take advantage of things that can be re-used in Drupal
 - Display modes
 - Blocks
 - Paragraphs
 - Layouts
- Lose out on rapid prototyping advantages.

STANDARD DRUPAL COMPONENT




```
<article{{ attributes.addClass(classes) }}>
```

```
  {{ title_prefix }}
```

```
  {% if label and not page %}
```

```
    <h2{{ title_attributes.addClass('title') }}>
```

```
      <a href="{{ url }}" rel="bookmark">{{ label }}</a>
```

```
    </h2>
```

```
  {% endif %}
```

```
  {{ title_suffix }}
```

```
<div{{ content_attributes.addClass('node__content') }}>
```

```
  {% if content.field_platform or content.field_year %}
```

```
    <div class="nes-badge is-splited">
```

```
      {% if content.field_platform %}<span class="is-dark">{{ content.field_platfo
```

```
      {% if content.field_year %}<span class="is-success">{{ content.field_year }}
```

```
    </div>
```

```
  {% endif %}
```

```
  {{ content|without('field_platform', 'field_year') }}
```

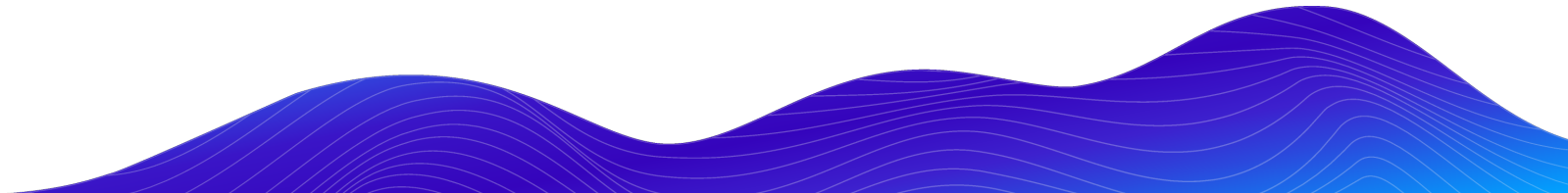
```
  <a class="nes-btn is-primary" href="{{ url }}" rel="bookmark">Press Start</a>
```

```
</div>
```

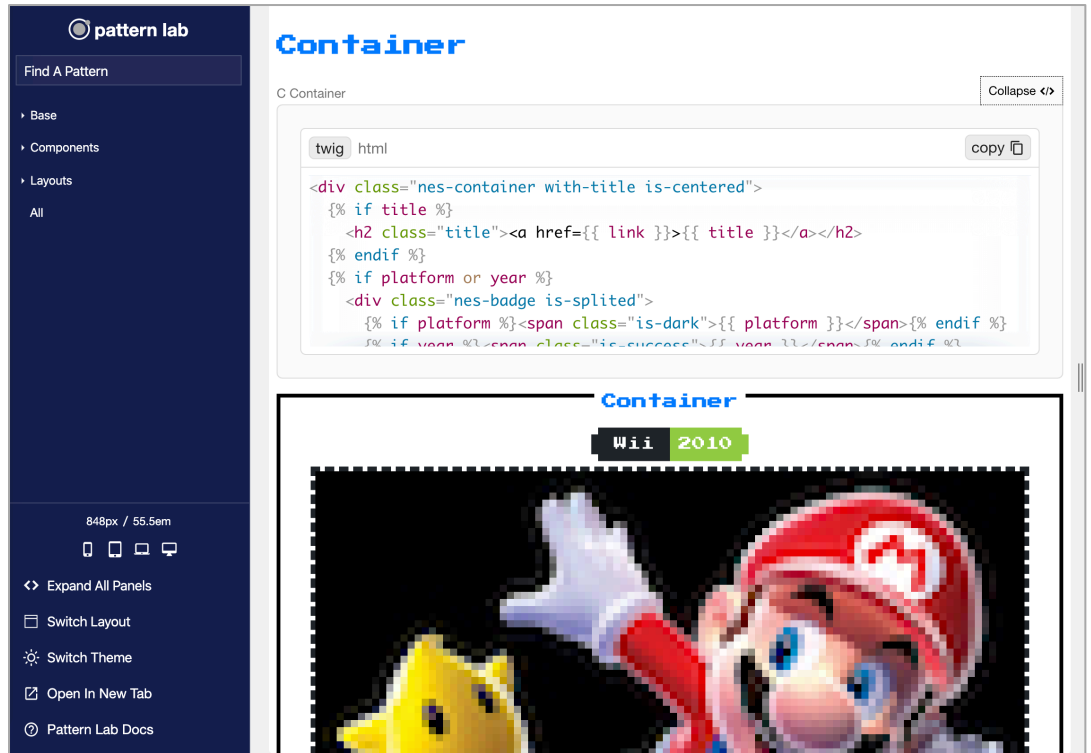
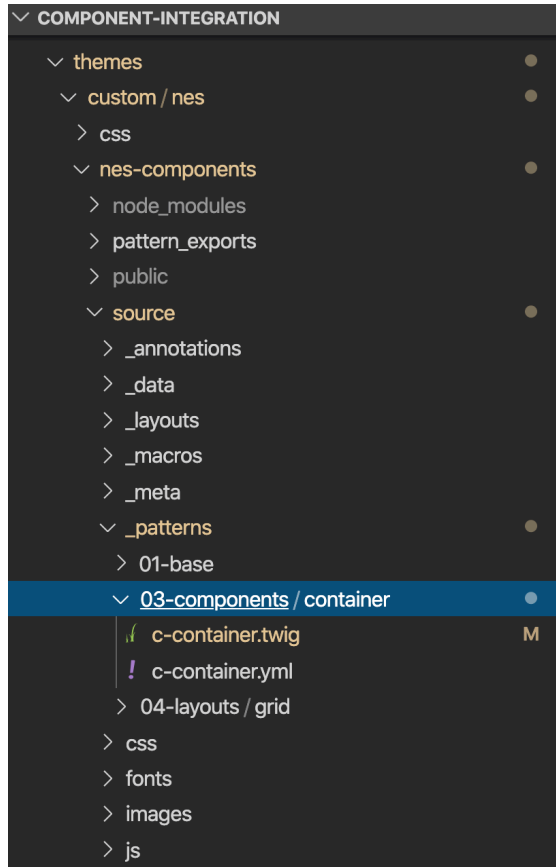
```
</article>
```

INTEGRATED DRUPAL COMPONENTS

Building components that live outside of the traditional templates directory



COMPONENT LIBRARY / PATTERN LAB



COMPONENTS MODULE

Creates Twig namespaces to access templates in non-standard locations

! *nes.info.yml* *web/themes/custom/nes/nes.info.yml*

```
component-libraries:
  components:
    paths:
      - nes-components/source/_patterns/03-components
```

INTEGRATION APPROACHES

Primarily fall into two categories

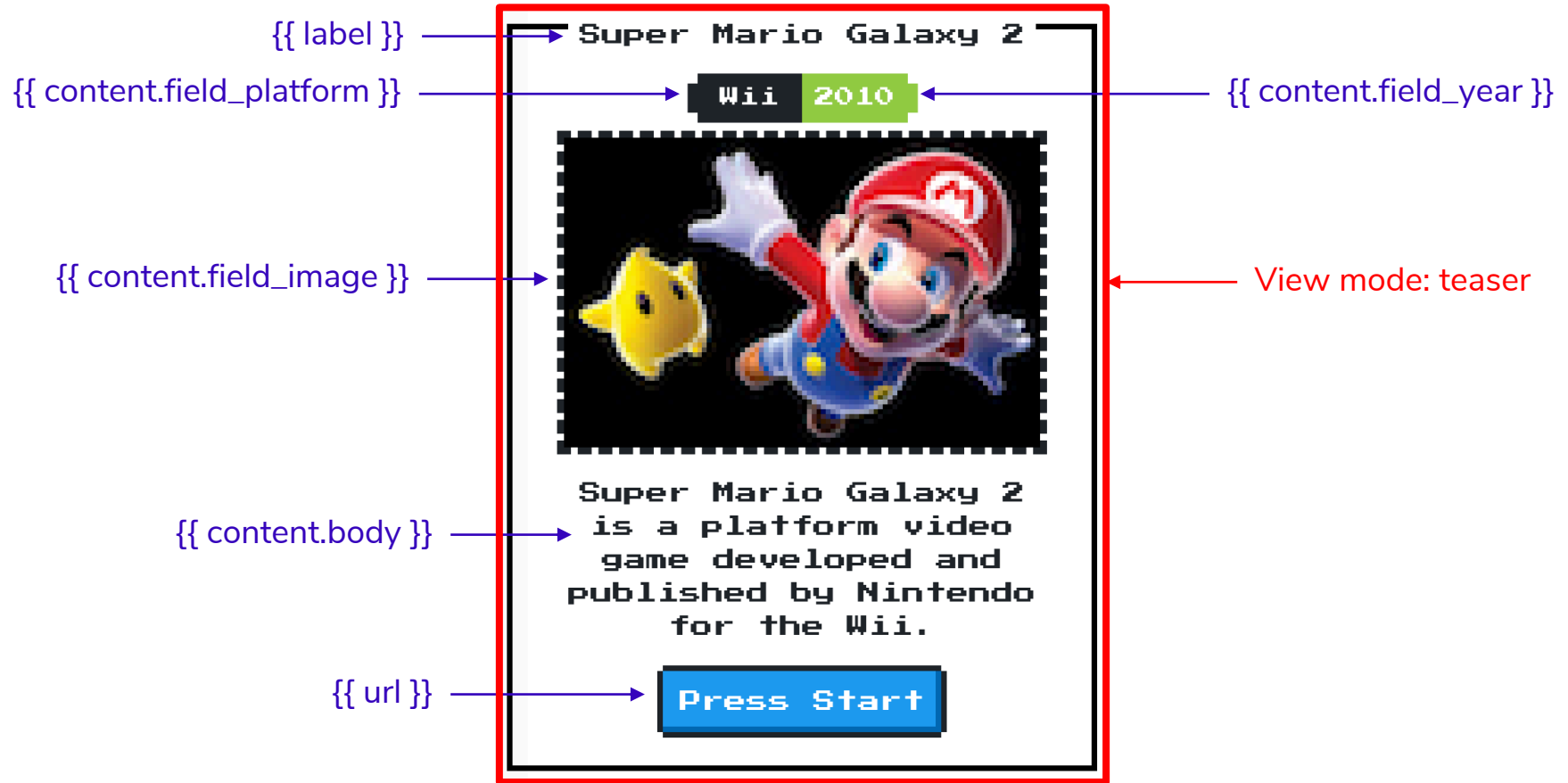
Mapping Data In Code

- Includes:
 - Mapping in Twig templates
 - Preprocessing
- More likely to get out of sync with Drupal UI
- More likely to break things like caching, Drupal functionality, etc.

Mapping Data In Admin UI

- Includes:
 - UI Patterns
 - Layouts
- Less likely to disrupt Drupal functionality
- Potentially not as flexible

INTEGRATING IN CODE



MAPPING IN TWIG PRESENTER TEMPLATE


Drupal template references template in component library

node--game--teaser.html.twig web/themes/custom/nes/templates/node--game--teaser.html.twig

```
<article{{ attributes.addClass(classes) }}>
  {% include "@components/container/c-container.twig"
    with {
      'title': label,
      'link': url,
      'platform': content.field_platform,
      'year': content.field_year,
      'image': content.field_image,
      'body': content.body
    }
  %}
</article>
```

DATA MAPPING IN PREPROCESS

Map in preprocess...

 nes.theme web/themes/custom/nest/nest.theme

```
/**
 * Implements hook_preprocess_node().
 */
function nes_preprocess_node__game(array &$variables) {
  $variables['title'] = $variables['label'];
  $variables['link'] = $variables['url'];
  $variables['platform'] = $variables['content']['field_platform'];
  $variables['year'] = $variables['content']['field_year'];
  $variables['image'] = $variables['content']['field_image'];
  $variables['body'] = $variables['content']['body'];
}
```

DATA MAPPING IN PREPROCESS

...and use simpler include in Twig presenter template

```
✓ node--game--teaser.html.twig web/themes/custom/nes/templates/node--game--teaser.html.twig
{
  {
    {%
      set classes = [
        'node',
        'node--type-' ~ node.bundle|clean_class,
        node.isPromoted() ? 'node--promoted',
        node.isSticky() ? 'node--sticky',
        not node.isPublished() ? 'node--unpublished',
        view_mode ? 'node--view-mode-' ~ view_mode|clean_class,
      ]
    %}
    {{ attach_library('classy/node') }}
    <article{{ attributes.addClass(classes) }}>
    {
      {% include "@components/container/c-container.twig" %}
    }
    </article>
  }
}
```

HELPER MODULES

Simplify Twig Mapping

Twig Field Value

- Get Partial data from field render arrays
 - field_label
 - field_value
 - field_raw
 - field_target_entity
- Map just the data you want
- May require additional caching considerations...

Twig Tweak

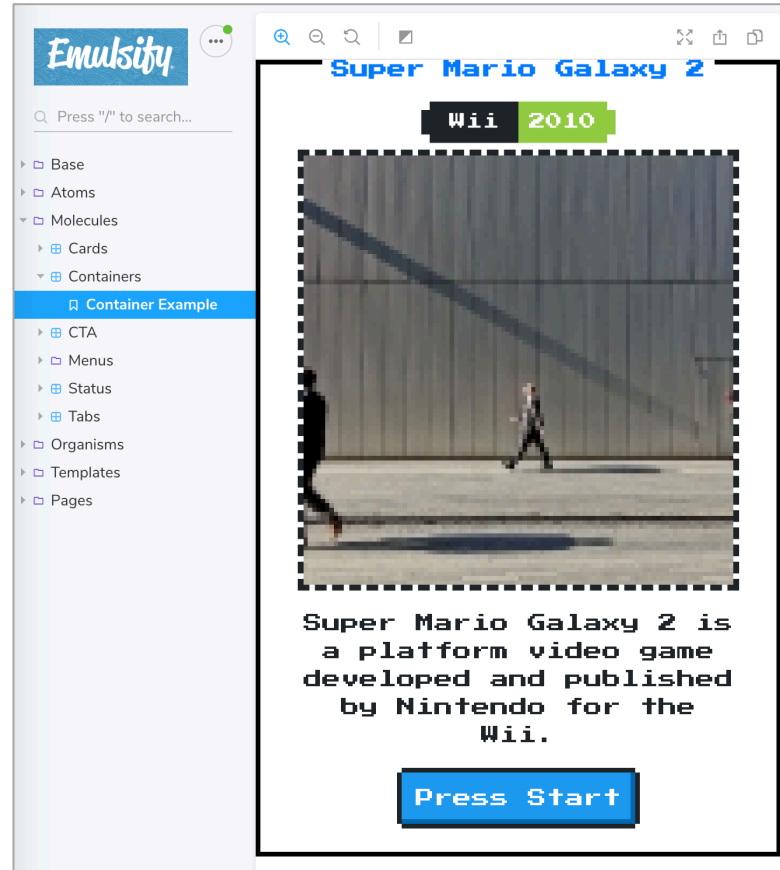
- Helpful twig functions and filters
 - Render views, blocks, regions, fields, entities and so on.
 - Render image with specific image style
 - Extract tokens from context

STARTER KITS AND THEMES

- Simplify set up and provide default tooling
- Some provide default components and helper functions
- Various levels of opinionated
- Examples:
 - Emulsify
 - Gesso
 - Shila
 - Particle

EMULSIFY DESIGN SYSTEM EXAMPLE

- Same presenter templates
- Different component location
- Different component library tool



JS container.stories.js web/themes/custom/nes_emulsify/components/02-molecules/container/container.stories.js/...



```
import React from 'react';
```

```
import container from './c-container.twig';
```

```
import './nes.min.css';
```

```
import './fonts.css';
```

```
import containerData from './c-container.yml';
```

```
/**
```

```
 * Storybook Definition.
```

```
 */
```

```
export default { title: 'Molecules/Containers' };
```

```
export const containerExample = () =>
```

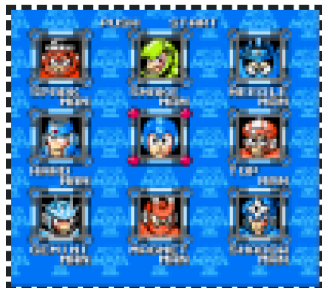
```
  <div dangerouslySetInnerHTML={{ __html: container(containerData) }} />;
```




Games

Mega Man 3

NES 1990



Mega Man 3 is an action-platform video game developed and published by Capcom.

Press Start

Metroid Prime

GC 2002



Metroid Prime is the fifth main installment in the Metroid series, and the first Metroid game played from the first-person perspective.

Press Start

Super Mario Galaxy 2

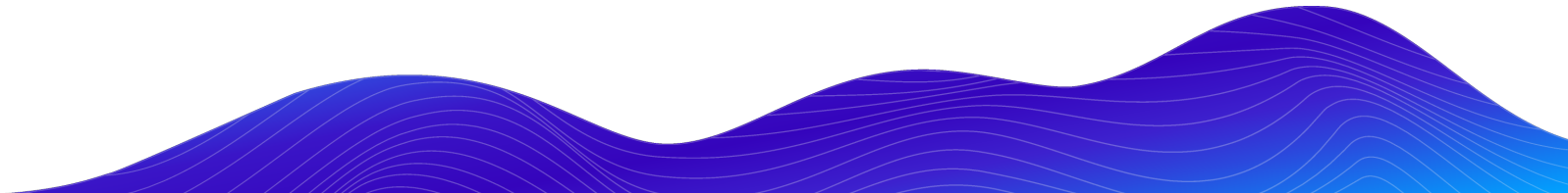
Wii 2010



Super Mario Galaxy 2 is a platform video game developed and published by Nintendo for the Wii.

Press Start

MAPPING DATA IN THE ADMIN UI



UI PATTERNS MODULE

Define and manage components in a way that Drupal understands

- Define UI Patterns as Drupal Plugins
- Configure data mappings in the UI
- Optional Pattern Library page exposed in Drupal
- Also allows Drupal to:
 - Preprocess patterns
 - Render patterns programmatically



```
container:
  label: Container
  description: A container component from NES.css
  fields:
    title:
      type: text
      label: Title
      description: Game title.
      preview: Super Mario Galaxy 2
    platform:
      type: text
      label: Platform
      description: Console or platform
      preview: Wii
    year:
      type: text
      label: Year
      description: Year of release
      preview: 2010
    image:
      type: image
      label: Image
      description: Box art
      preview: Super Mario Galaxy 2 is a platform video game developed and publis
    link:
      type: url
      label: Link
      description: link to node
      preview: '#'
```

```
    preview: <p>Super Mario Galaxy 2 is a platform video game developed and published by Nintendo.  
link:  
  type: url  
  label: Link  
  description: link to node  
  preview: '#'  
libraries:  
  - pattern_library_one:  
    css:  
      component:  
        ../../nes-components/source/css/c-container.css: {}  
use: "@components/container/c-container.twig"
```

```
    preview: <p>Super Mario Galaxy 2 is a platform video game developed and published by Nintendo.  
link:  
  type: url  
  label: Link  
  description: link to node  
  preview: '#'  
libraries:  
  - pattern_library_one:  
    css:  
      component:  
        ../../nes-components/source/css/c-container.css: {}  
use: "@components/container/c-container.twig"
```

Pattern library

Available patterns

[. Container](#)

Container

A container component from NES.css

Field	Label	Type	Description
title	Title	text	Game title.
platform	Platform	text	Console or platform
year	Year	text	Year of release
image	Image	image	
body	body	text	Body text
link	Link	url	link to node

Preview

[Super Mario Galaxy 2](#)

Wii 2010



UI PATTERNS VIEWS

Games (Content) ☆

[Home](#) » [Administration](#) » [Structure](#) » [Views](#)

Displays

Page

+ Add

Edit view name/description ▼

Display name: [Page](#)

View Page ▼

TITLE

Title: [Games](#)

FORMAT

Format: [Unformatted list](#) | [Settings](#)

Show: [Pattern](#) | [Settings](#)

FIELDS

Content: [Title](#)

Content: [Body](#)

Content: [Image](#)

Content: [Link to Content](#)

Content: [Platform](#)

Content: [Year](#)

ADD ▼

PAGE SETTINGS

Path: [/games](#)

Menu: [No menu](#)

Access: [Permission](#) | [View published content](#)

HEADER

Add

FOOTER

Add

NO RESULTS BEHAVIOR

Add

PAGER

Use pager: [Mini](#) | [Mini pager, 10 items](#)

More link: [No](#)

▶ ADVANCED

bounteous







36

Page: Row style options

- ☐ Provide default field wrapper elements
- If not checked, fields that are not configured to customize their HTML elements will get no wrappers at all for their field, label and fieldset. You can use this to quickly reduce the amount of markup the view provides by default, at the cost of making it more difficult to apply custom CSS.
- ☒ Hide empty fields
- Do not display fields, labels or markup for fields that are empty.

Pattern *

Container ▾

SOURCE	PLUGIN	DESTINATION
 Content: Title	Views row	Title ▾
 Content: Body	Views row	body ▾
 Content: Image	Views row	Image ▾
 Content: Link to Content	Views row	Link ▾
 Content: Platform	Views row	Platform ▾
 Content: Year	Views row	Year ▾

Apply

Cancel



LAYOUTS



LAYOUTS AND LAYOUT BUILDER

! nes.layouts.yml web/themes/custom/nes/nes.layouts.yml

```
container:
  label: 'Container'
  category: 'NES'
  template: templates/container-layout
  default_region: body
  regions:
    title:
      label: Title
    platform:
      label: Platform
    year:
      label: Year
    image:
      label: Image
    body:
      label: Body
    link:
      label: Link
```

container-layout.html.twig web/themes/custom/nes/templates/container-layout.html.twig

```
{% include "@components/container/c-container.twig"
with {
  'title': content.title,
  'link': content.link,
  'platform': content.platform,
  'year': content.year,
  'image': content.image,
  'body': content.body
}
%}
```

LAYOUTS AND LAYOUT BUILDER

```
c-container.twig web/themes/custom/nes/nes-components/source/_patterns/03-components/container/c-contain
<div {{ attributes.addClass('container') }}>
  <div class="nes-container with-title is-centered">
    {% if title %}
      <div {{ region_attributes.title.addClass('title-region') }}>
        <h2 class="title"><a href={{ link }}>{{ title }}</a></h2>
      </div>
    {% endif %}
    {% if platform or year %}
      <div class="nes-badge is-splited">
        {% if platform %}<span class="is-dark">{{ platform }}</span>{% endif %}
        {% if year %}<span class="is-success">{{ year }}</span>{% endif %}
      </div>
    {% endif %}
    {% if image %}
      <div {{ region_attributes.image.addClass('image-region') }}>
        {{ image }}
      </div>
    {% endif %}
    {% if body %}
      <div {{ region_attributes.body.addClass('body-region') }}>
        {{ body }}
      </div>
    {% endif %}
    {% if link %}
      <a class="nes-btn is-primary" href={{ link }}>Press Start</a>
    {% endif %}
  </div>
</div>
```


COMPONENT BLOCKS

Best of both worlds

- Recently released (stable release, but early)
- Exposes UI Patterns to Layout Builder
- Sidesteps visual layout issues
- Use any fields available to the entity, along with fixed inputs.

Component blocks

[View](#) [Version control](#) [View history](#) [Automated testing](#)

By [larowlan](#) on 15 July 2020, updated 15 July 2020


[★ Unstar](#) 10 [✉ Followed](#)

Summary

Provides integration between the [UI Patterns](#) module and core's Layout Builder.

Features

One block derivative is created per UI pattern.
The configuration form for the block lets you pick fields from the entity in scope (e.g. the current node) as well as which formatter to use for each field.
You can also use a fixed string, with token support.



Maintainers for Component blocks

[larowlan](#) – 3 commits
last: 3 weeks ago, first: 3 weeks ago
[View all committers](#)
[View commits](#)

Issues for Component blocks

To avoid duplicates, please search before submitting a new issue.

[Search](#)

[Advanced search](#)
All issues
[2 open](#), [3 total](#)
Bug report
[0 open](#), [0 total](#)
Statistics

New issues

Response rate

1st response

0

0 %

0 hours

COMPONENT BLOCKS

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 Game content items.



 You have unsaved changes.

+Add section

 [Configure Container Section](#)

+Add block

+Add section

 Choose a block 

 [Create custom block](#)

Filter by block name

- ▼ Component blocks
- Container with fields from Content

Container with fields from User

- ▼ Content fields
- Authored by

Authored on

Body

Changed

Content type

Default revision

Default translation

ID

Image

Language

Links

Platform

Promoted to front page

Published

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 Game content items.

⚠️ You have unsaved changes.

+Add section

✕ [Configure Container Section](#)

"Container" block

+Add block

✎ Configure block ✕

Block description Container with fields from Content

Title *

Container

☐ Display title

▼ Context variables

Title

Source

Title

Formatter *

Plain text

☒ Link to the Content

Platform

Source

Fixed input

Fixed value

Year

Source

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 Game content items.

+Add section

⌵ [Configure Container Section](#)

"Container" block

+Add block

+Add section

✎ Configure block ✕

Year

Source

Year

Formatter

Plain text

☐ Link to the Content

Image

Source

Image

Formatter

Image

Image style

Pixelated

[Configure Image Styles](#)

Link image to

Nothing

Body

Source

Body

COMPONENT

Not to be confused with 'Components'...

- Recently released
- Another way to define a component via yml
- Can derive block configuration
- More focused on Decoupled use case (inspired by PDB)

Component

[View](#) [Version control](#) [View history](#) [Automated testing](#)

By [rlnorthcutt](#) on 6 May 2007, updated 31 May 2020

This project is not covered by Drupal's [security advisory policy](#).

Adding JS components to your Drupal site just got a whole lot easier. Just combine your JS components (any type) with an `info.yml` and put it in the codebase. Now, your component will be available in Drupal as a block – automatically!

You can also add a configuration form to your component so site builders can modify the component. This component "looks" like any other block, so it can be used just like a core block.

JS devs don't need to know PHP or Drupal in order to integrate their components into the CMS.

They just need to setup the `info.yml` file properly. The `info.yml` file provides the JS developer with a ton of basic configuration options. By modifying this file, you can provide static or dynamic parameters, include additional libraries, and even adjust the cache configuration. See the code comments on `example_tabs.info.yml` for details.

Example Config

name	Leonard McCoy
greeting	Dr
active	on
interests	music,sports
favcolor	#ff2600
date	2227-01-20

Configure Block

Block description: Example Config

Title: Machine name: exampleconfig

☐ Display title

COMPONENT SETTINGS

Name:

Greeting:

☒ Active

Interests: ☒ Sports

Art: ☐ Games: ☐ Books: ☐

Favourite color:

Visibility:

```
name: example_config
machine_name: example_config
type: component
description: 'Example component showing the new config options available'
core_version_requirements: '^8 || ^9'
modules: ['component_example']
js:
  example_config.js: {}
  styles:
```

! evolutionnavbar.component.yml docroot/themes/custom/bo

You, seconds ago | 2 authors (Wade Stewart and others)

name: Evolution Navbar

description: 'Evoution Navbar'

type: 'block'

js:

'dist/navbar.bundle.js' : {}

dependencies:

- bounteous/react

template: 'evolutionnavbar.html'

form_configuration:

theme:

type: select

title: "Theme"

options:

'': 'Dark Theme'

'theme-light': 'Light Theme'

default_value: ''

JS index.js gatsby/src/exports/EvolutionNavbar/index.js/...

Wade Stewart, a month ago | 2 authors (You and others)

import React from "react"

import ReactDOM from "react-dom"

import { MediaContextProvider } from "../../components/layouts/Media/Media"

import Navbar from "../../components/components/Navbar/Navbar"

import abstracts from "styles/abstracts.scss"

const drupalProvider = document.querySelector(".evolutionnavbar");

const config = drupalProvider.dataset;

ReactDOM.render(

<MediaContextProvider>

<Navbar color={abstracts.cPlum} theme={config.theme} />

</MediaContextProvider>,

document.getElementById("evolution-navbar")

)

evolutionnavbar.html is just:

<div id="evolution-navbar"></div>

You, 2 m

RENDER COMPONENT AS COMPONENT BLOCK

Settings passed as data attributes
on wrapping div.

Configure block

Configure blockTranslate blockDevel

[Admin](#) » [Structure](#) » [Block](#)

Block description: Evolution Navbar

Title *

Evolution Navbar

Machine name:

This field supports tokens. [Browse available tokens.](#)

☐ Display title

▼ **COMPONENT SETTINGS**

Theme

Dark Theme ▼

Visibility

Content type

Webform

[Content types](#)
Not restricted

[Pages](#)
Not restricted

Content type

☐ Article

☐ Blog

☐ Case Study

☐ Client

☐ Client Quote

☐ Client Resource Page

☒ Client Story

☐ Components Page

COMPONENT DEFINITION APPROACHES

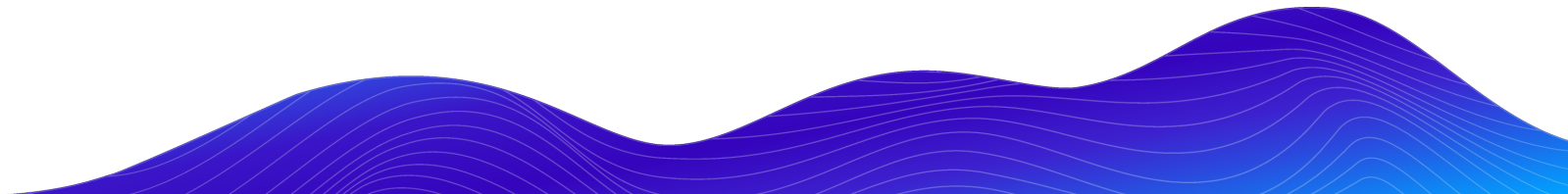
Manual Definition

- Define component in code so that Drupal becomes aware of it.
- Likely requires some amount of duplication between Drupal and component library

Automatic Discovery

- Drupal module automatically discovers components from component library and makes them available to Drupal.
- Emerging/experimental concept.
 - Dev modules, proceed with caution.
- Expects a particular convention and thus won't work with all component libraries.

AUTOMATIC DISCOVERY



UI PATTERNS PATTERN LAB


Automatically create UI Patterns from your pattern library... really.

- End result same as previous UI Patterns Example
- No redundant ui_patterns.yml file necessary
- Some limitations
 - Requires yaml or json file with pattern data
 - Requires specific approach to nested components.

UI Patterns Pattern Lab

[View](#) [Version control](#) [Automated testing](#)

Posted by [brianperry](#) on 1 May 2018, updated 7 May 2018

 This project is not covered by Drupal's [security advisory policy](#).

The UI Patterns Pattern Lab module automatically discovers patterns defined in a Pattern Lab instance and makes them available to be used in Drupal as [UI Patterns](#).

This module will recognize Pattern Lab patterns in any active module or theme's /templates directory, along with any paths defined as Twig Namespaces in your theme by the [Component Libraries](#) module. After enabling this module (which will also enable the dependencies `ui_patterns` and `ui_patterns_library`) and clearing your cache, patterns should be visible at `/patterns` and available to use with any of the UI Patterns integration modules.

This project would not exist without the work of [Antonio De Marco](#) who maintains the [UI Patterns](#) module and [Pierre Dureau](#) who created the [UI Patterns Fractal integration](#) that this project is based on.

UI

UI

UI

PatternLab

UI Patterns

PATTERNKIT

Combines aspects of manual definition and automatic discovery

- Requires creating schema definition file (which has potential applications outside of Drupal)
- Automatically derives blocks from pattern library components
- Supports a specific set of field types
- Token support in D7 but not yet D8

Patternkit

[View](#) [Version control](#) [View history](#) [Automated testing](#)

By [cyb.tachyon](#) on 30 January 2018, updated 7 February 2020

This project is not covered by Drupal's [security advisory policy](#).

Patternkit loads your templates/patterns into Drupal as blocks*, where you can add them to your pages and layouts. The fields are read from a similarly-named JSON Schema file next to the twig template.

This allows Drupal 8 Core's Layout Builder or anything that uses blocks to function as a full custom page builder app.

*Panels in Drupal 7.

This module will parse a pattern library (local or through REST endpoints) to generate a list of blocks that can be drag/dropped into layouts. It currently supports directories full of Twig templates, and support for additional pattern types is planned. All you need to do is add a "patterns" section to your *{theme-or-module}.libraries.yml*, and a .json file written with JSON Schema (Draft 4) to power the editor.

patternkit_example.libraries.yml

```
patternkit:  
  version: VERSION
```


PATTERNKIT PATTERN DISCOVERY

! nes.libraries.yml web/themes/custom/nes/nes.libraries.yml

global-styling:

version: 1.x

css:

theme:

nes-components/source/css/nes.min.css: {}

nes-components/source/css/c-container.css: {}

nes-components/source/css/l-grid.css: {}

css/drupal.css: {}

<https://fonts.googleapis.com/css?family=Press+Start+2P>: { type: external }

patterns:

nes-components/source/_patterns/03-components/container: {plugin: twig}

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "Container",
  "description": "NES Container",
  "category": "pattern",
  "type": "object",
  "properties": {
    "title": {
      "title": "Title",
      "type": "string"
    },
    "platform": {
      "title": "Platform",
      "type": "string"
    },
    "year": {
      "title": "Year",
      "type": "string"
    },
    "body": {
      "title": "Body",
      "type": "string",
      "format": "textarea",
      "options": {
        "wysiwyg": true
      }
    },
    "link": {
      "title": "Link",
      "type": "string"
    },
    "settings": {
      "title": "Settings",
      "type": "object",
      "properties": {
        "alignment": {
          "title": "Item Alignment",
          "type": "string",
          "enum": [
```

Configure block ☆

[Home](#) » [Administration](#) » [Structure](#) » [Block layout](#)

Block description: [Patternkit] Container

☐ Reusable

Presentation style

HTML inline ▼

Container ▼

 Properties

NES Container

Title

Super Mario Galaxy

Platform

Wii

Year

2010

Body

Super Mario Galaxy 2 is a platform video game developed and published by Nintendo for the Wii.

Link

#

Settings ▼

 Properties

Item Alignment

Default



Component Integration Sandbox

Pattern Kit Example

[View](#)

[Edit](#)

[Delete](#)

[Revisions](#)

Super Mario Galaxy

Wii 2010

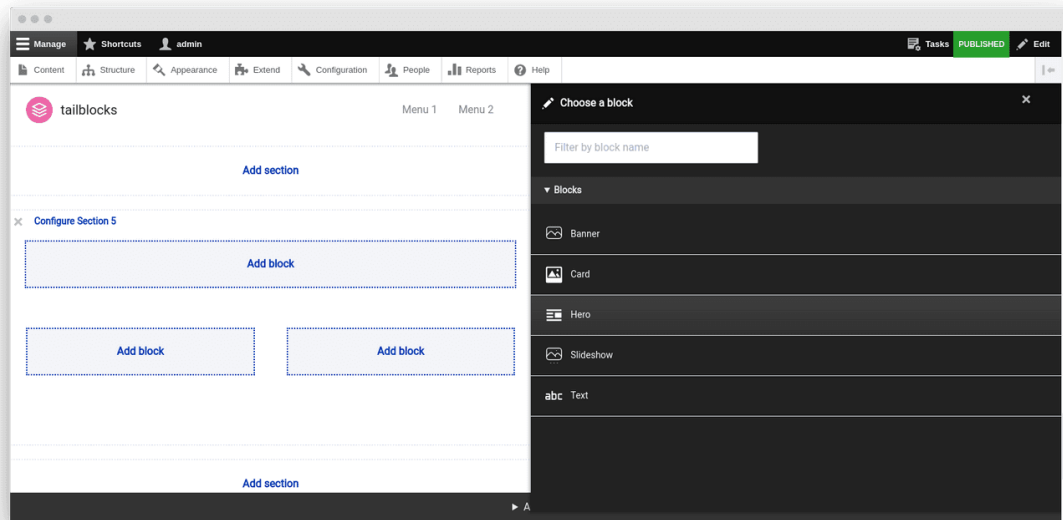
Super Mario Galaxy 2 is a platform video game developed and published by Nintendo for the Wii.

Press Start

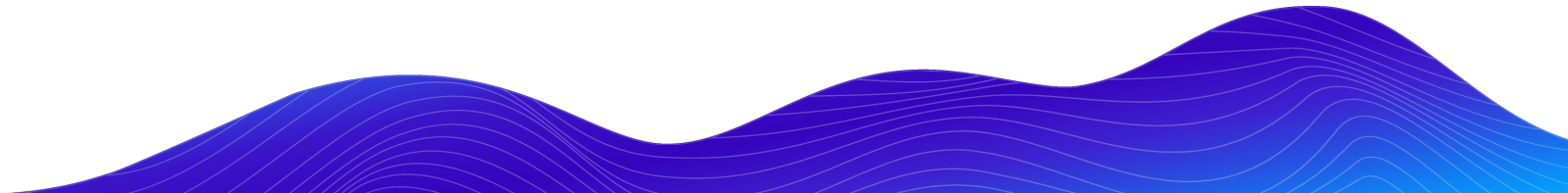
WINGSUIT

Tool to build Twig UI Components with Storybook

- Extended variants of UI Patterns
- Generate components via cli
- Use with zero config in Drupal with wingsuit_companion module
- Wingsuit Kickstarter project available



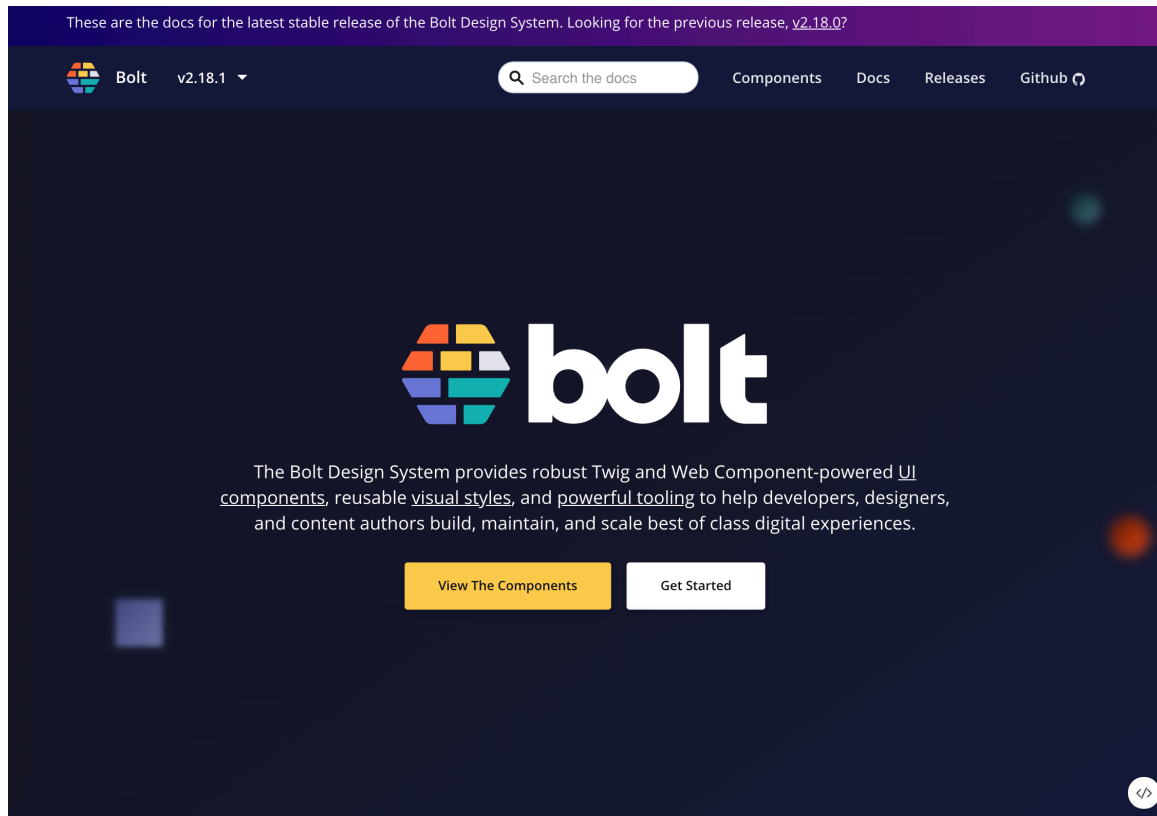
PRE-PACKAGED COMPONENT SOLUTIONS



BOLT DESIGN SYSTEM

Ready to use web-components

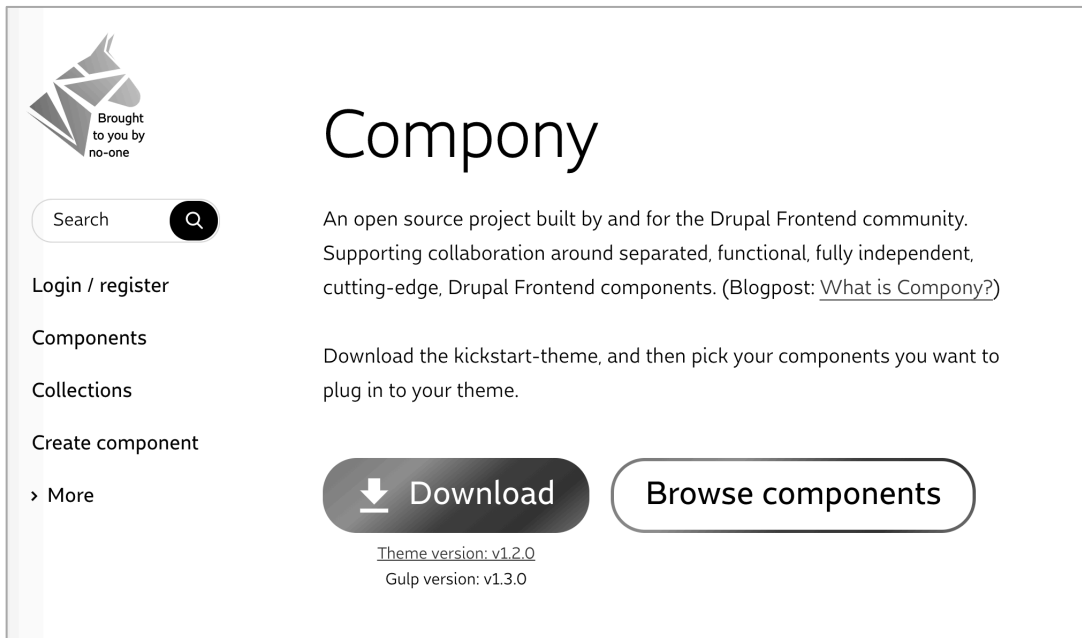
- Full design system
- Selectively require components.



COMPONY

Component distribution system

- Combines a theme, Gulp workflow and components.
- Download existing components or create your own
- Not Composer / NPM driven



The screenshot shows the Compony website. On the left is a sidebar with a logo that says 'Brought to you by no-one'. The sidebar contains links: 'Search' (with a magnifying glass icon), 'Login / register', 'Components', 'Collections', 'Create component', and '> More'. The main content area has the title 'Compony' in a large font. Below the title is a paragraph: 'An open source project built by and for the Drupal Frontend community. Supporting collaboration around separated, functional, fully independent, cutting-edge, Drupal Frontend components. (Blogpost: [What is Compony?](#))'. Below this is another paragraph: 'Download the kickstart-theme, and then pick your components you want to plug in to your theme.' At the bottom of the main area are two buttons: 'Download' (with a download icon) and 'Browse components'. Below the 'Download' button, it says 'Theme version: v1.2.0' and 'Gulp version: v1.3.0'.

SINGLE FILE COMPONENTS

Drupal components with Vue style syntax

- Use like any template
- Automatically generates library definitions
- Derive Blocks and Layouts with Annotations
- Provides component library
- Doesn't really solve integration problem
- Does help with distribution and re-use.

```
<?php

namespace Drupal\sfc_test\Plugin\SingleFileComponent;

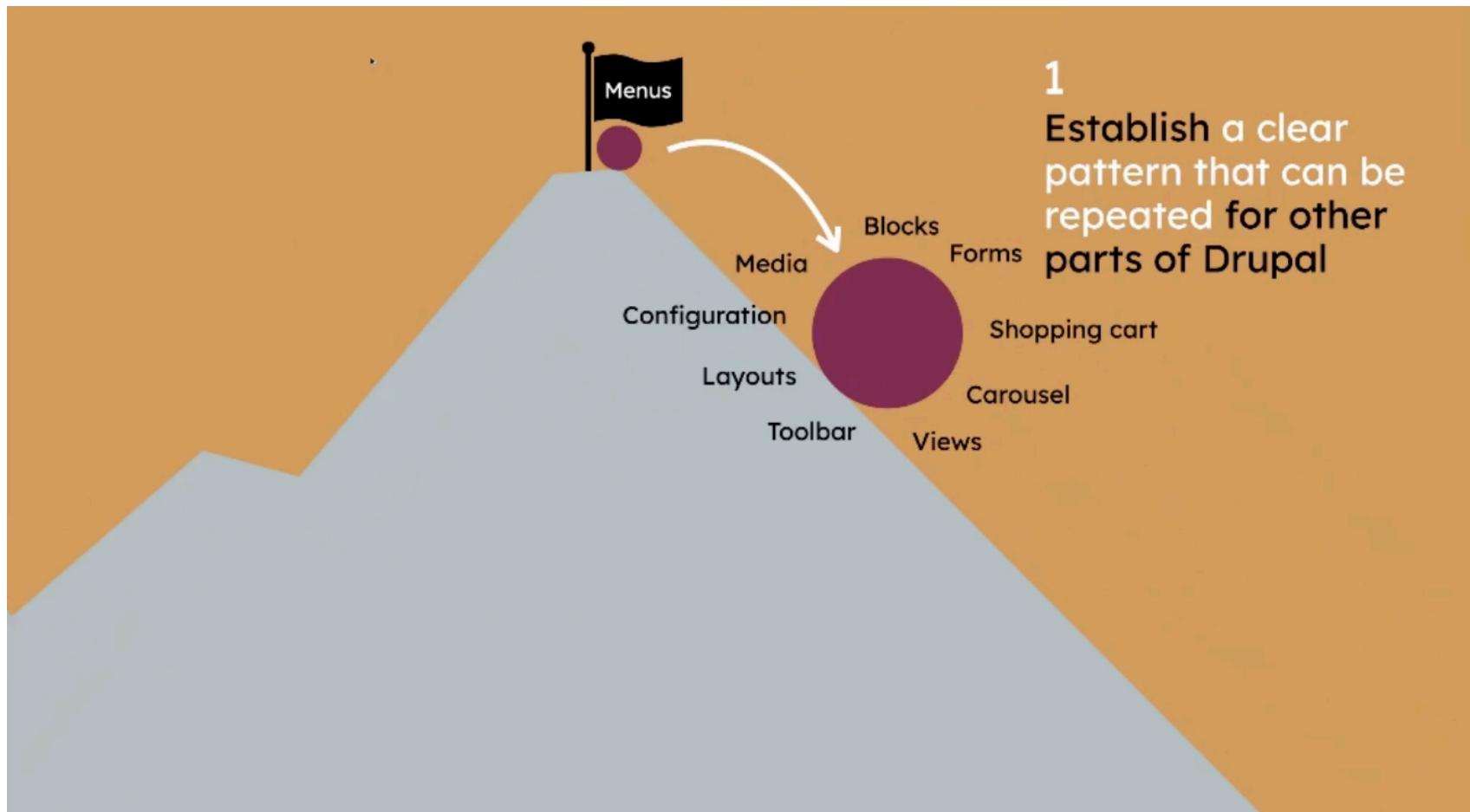
use Drupal\sfc\ComponentBase;

/**
 * Contains an example single file component.
 *
 * @SingleFileComponent(
 *   id = "say_hello",
 * )
 */
class SayHello extends ComponentBase {

    const TEMPLATE = <<<TWIG
    <p class="say-hello">Hello {{ name }}!</p>
    TWIG;

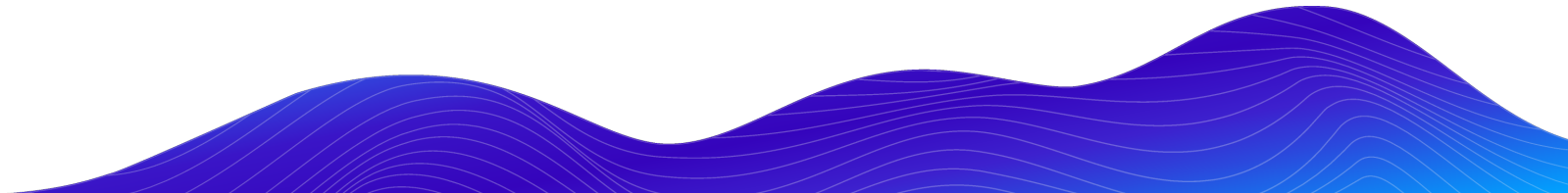
    const CSS = <<<CSS
    .say-hello {
        color: pink;
    }
    CSS;

    /**
     * {@inheritdoc}
     */
    public function prepareContext(array &$context) {
        if (!isset($context['name'])) {
            $context['name'] = \Drupal::currentUser()->getDisplayName();
        }
    }
}
```

COMPONENT WORKFLOW

Present and Future



CURRENT APPROACH(ES)

Leveraging a mix of approaches

- Integrated components in a custom theme
 - Majority of twig/sass/js inside Pattern Library instance.
- Defining component mapping in Drupal UI for lightweight components
- Preprocess for components with heavy logic
 - Created preprocess helper abstract class – hope to open source in future
 - Project specific helper functions
 - Limited mapping in twig templates
- Build components compatible with Layout Builder
 - Custom block types with limited use of Paragraphs

SO... WHAT SHOULD I DO?

Start by mapping in code.

- Most battle tested approach
- Chose an approach based on team makeup
 - FE focused – Twig mapping
 - BE savvy – Preprocess
- Consider other methods as experience grows

DREAM WORKFLOW

Basically React (or insert the name of your favorite JS framework here)

- Build fully packaged distributable components
- Easily install them
 - `npm install cool-component / composer require drupal/cool-component`
- Import them in code
 - `import 'CoolComponent' from 'cool-component'; / {% include '@components/cool-component.twig' %}`
- Use them as I see fit
 - `<CoolComponent />`

HOW DO WE GET THERE?

Have a lot of the pieces, but need a little extra ‘magic’

- Make it easier to package, distribute and use individual components
 - Track evolution of Web Components
- Improve UI based component configuration process in Drupal
 - With specific focus on Layout Builder.
 - Component Blocks seems to bridge this gap well
- Evolve approaches allowing Drupal to automatically discover components
- Keep building amazing looking component based sites using Drupal

Thanks to the many **Drupal**
component ecosystem
contributors!

Join us for contribution opportunities

Friday, December 11, 2020

Mentored
Contribution
9h00-18h00 CET

First Time
Contributor Workshop
10h00-12h00 CET

General
Contribution
9h00-18h00 CET

#DrupalContributions

What did you think?

Take time to complete the survey at the end of this session



Thank you!

Q&A

Brian Perry

Lead Front End Developer

Email: brian.perry@bounteous.com

