NTT DATA

DRUPAL IBERIA

Drupal's Secret Weapon: Single Directory Components and Its Supporting Cast

Speakers Information

Sara Barroso

Drupal Lead Engineer @ NTT DATA Portugal

CONTACT

sara.alexandra.barroso@nttdata.com

/u/sara_asb on drupal.org

BIOGRAPHY

Drupal developer
Acquia Certified Drupal 9 Site Builder
Acquia Certified Drupal 9 Developer

LinkedIn

Speakers Information

Catarina Aleixo Piza

Drupal Lead Engineer @ NTT DATA Portugal

CONTACT

catarina.aleixo.piza@nttdata.com

/u/catarina-piza on drupal.org

BIOGRAPHY

Drupal developer
Acquia Certified Drupal 10 Developer
Acquia Certified Drupal 8 Site Builder

Linkedin

Single Directory Components: Creators

Mateu Aguiló Bosch
(e0ipso)

Mike Herchel
(mherchel)

Lauri Eskola
(lauriii)

Agenda

1. Journey & Obstacles of a Drupal Frontend Developer
2. Single Directory Components (SDC) solution overview
3. Common mistakes and best practices
4. The Power of Integration: SDC with Complementary Modules
5. Benefits of Using Complementary Modules with SDC
6. Future Perspectives: Predicting the Evolution and Impact of SDC in the Drupal Community

NTT DATA

01

Journey & Obstacles of a Drupal Frontend Developer

- Developing a new “component”
- Debugging

Journey of a Drupal Frontend Developer

Start

After modelling the requirements, and creating the necessary entities, view modes, etc.

Starting work on a "component".

Finding the necessary template files.

Attaching the necessary libraries.

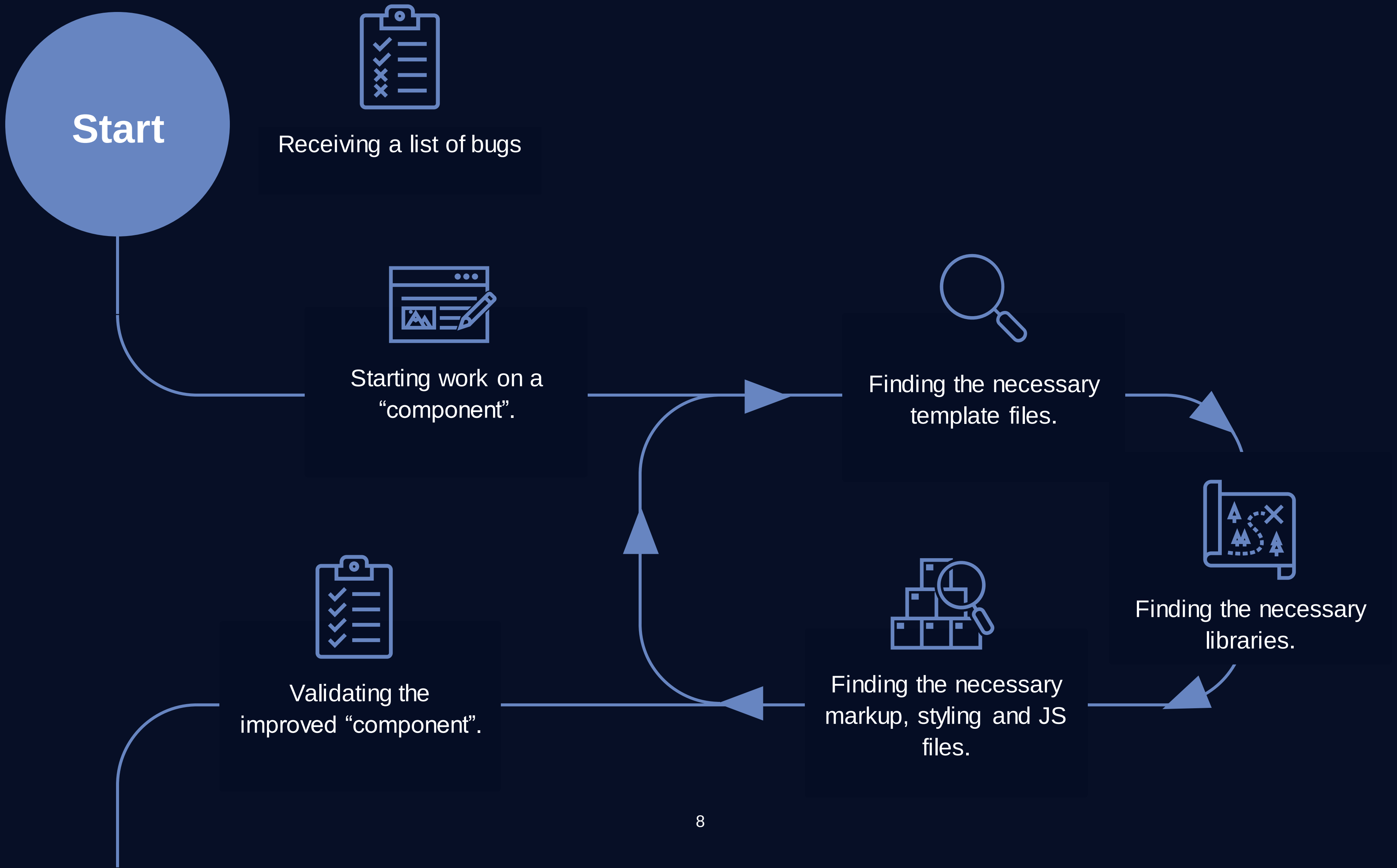
Coding the necessary markup, styling and JS.

Validating the new "component".

Repeat for all site "components"

(even for variations of previously developed ones)

Journey of a Drupal Frontend Developer: Debugging


Start
Receiving a list of bugs
Starting work on a "component".
Finding the necessary template files.
Finding the necessary libraries.
Finding the necessary markup, styling and JS files.
Validating the improved "component".

Obstacles: Developers with different backgrounds

Drupalisms

- Render pipelines
- Attaching libraries
- …

Scattered files

- Templates
- Styles
- Assets
- Modules

Obstacles: Developers with experience in Drupal

Repetition within the same project

- Same behavior referenced in different pages
- Make code general enough (could be heavy)
- Repeat code

Repetition across different projects

- Most projects will use similar behaviors:
  - Dropdowns
  - Carrousels/Sliders
  - Accordions, etc.

02

Single Directory Components: solution overview

- What is it?
- Why use it?
- Example implementation

Single Directory Components (SDC): What is it?

Module

- Module SDC now in core >= 10.1
- Allows the development of modular and reusable components

Primary goal

- “Simplify the front-end development workflow and improve the maintainability of custom, Core, and contrib themes.”

Philosophy

- Components are like building blocks with which we create web pages
- Components are typically made up of HTML, CSS, and JavaScript code
- Within SDC, all files necessary to render a component are grouped together in a single directory

Single Directory Components (SDC): Why use it?

Benefits

- Organization
- Automatic library creation
- Reusability
- Consistency
- Scalability
- Testing
- Collaboration

Other advantages

- Abstraction from Drupal specific theming concepts
- Any module and theme can provide components and can be overridden within your theme
- Available as an (experimental) module in Drupal core
- Components are plugins, automatically discovered by Drupal if placed within the correct directory

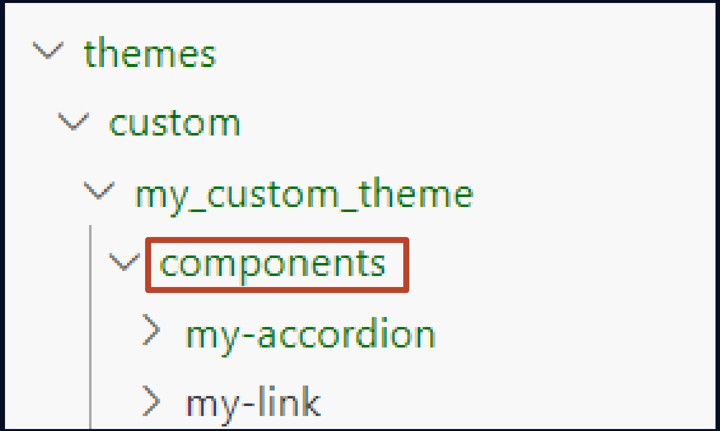
SDC: Example implementation

Before you start

- Make sure you are using Drupal 10.1 or greater
- Make sure you enable the Single Directory Components module
- You must have a theme or module in which to add components.
- If it is a theme, it must be installed

Established structure

- Content type "Article" has a paragraph field
- Paragraph "Accordion" has two fields:
  - Title – plain text
  - Description – plain text (long)

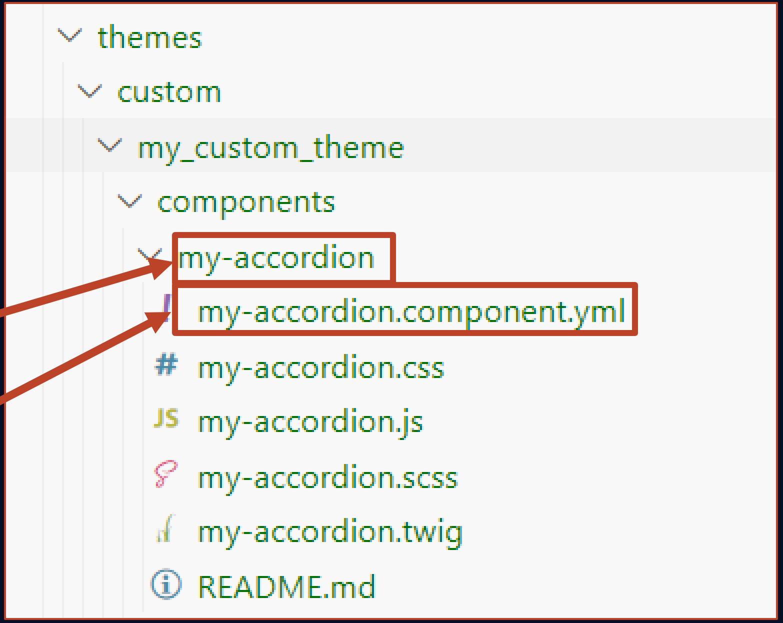


```
themes
  custom
    my_custom_theme
      components
        my-accordion
        my-link
```

SDC: Example implementation

Define the component

- Pick a name (it should be unique within the theme) – in this example we will be working with my-accordion
- Create the directory /my-accordion
- Create my-accordion.component.yml
- Populate the .component.yml file with metadata that describes the component



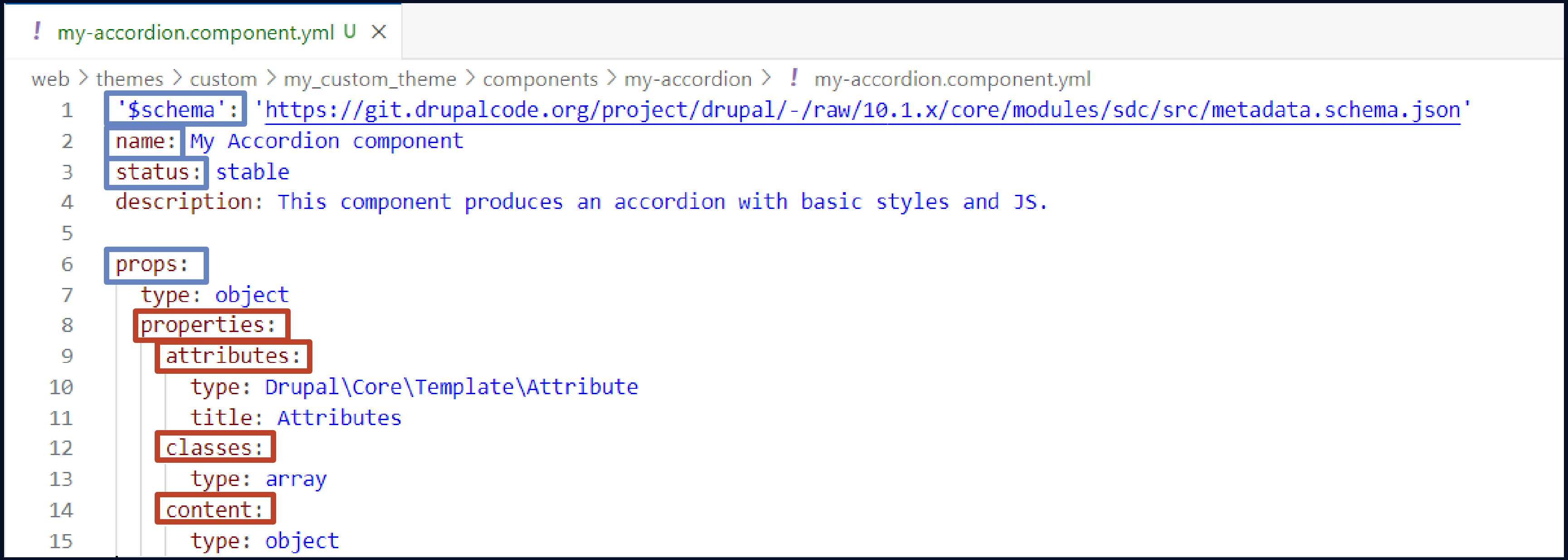
```
themes
  custom
    my_custom_theme
      components
        my-accordion
          my-accordion.component.yml
          my-accordion.css
          my-accordion.js
          my-accordion.scss
          my-accordion.twig
          README.md
```

See the [annotated example component]() for example metadata

SDC: Example implementation

my-accordion.component.yml keys (all optional):

- $schema – so the IDE knows about the syntax
- name – human readable name
- status – "experimental", "stable", "deprecated", "obsolete“
- props – allows for the use of variables whose type and structure is well defined
- slots – allow for the passing data with an unknown structure

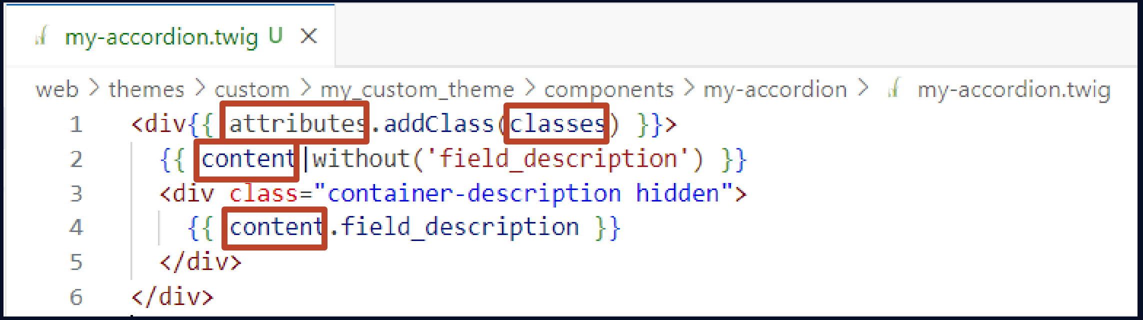

! my-accordion.component.yml U ×
web > themes > custom > my_custom_theme > components > my-accordion > ! my-accordion.component.yml

```yaml
'$schema': 'https://git.drupalcode.org/project/drupal/-/raw/10.1.x/core/modules/sdc/src/metadata.schema.json'
name: My Accordion component
status: stable
description: This component produces an accordion with basic styles and JS.

props:
  type: object
  properties:
    attributes:
      type: Drupal\Core\Template\Attribute
      title: Attributes
    classes:
      type: array
    content:
      type: object
```

SDC: Example implementation

Adding markup, styling and js files

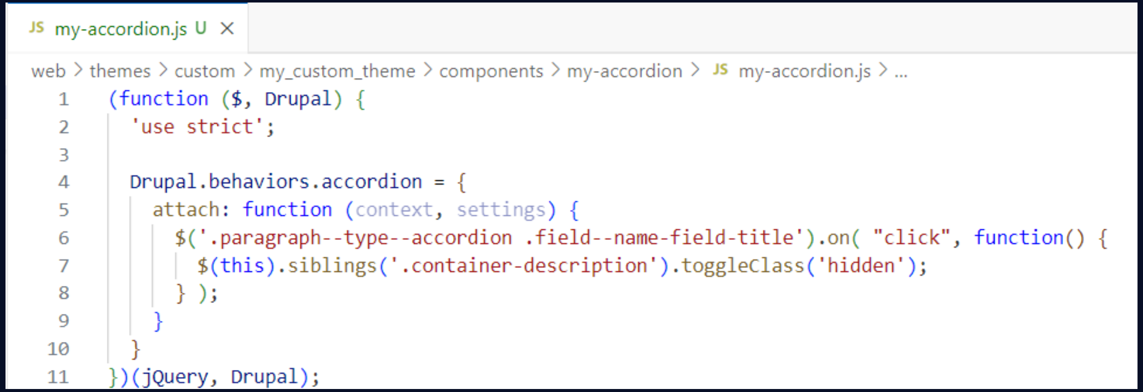
- Create my-accordion.twig
- Create my-accordion.css
- Create my-accordion.js



my-accordion.twig U ×

web > themes > custom > my_custom_theme > components > my-accordion > my-accordion.twig

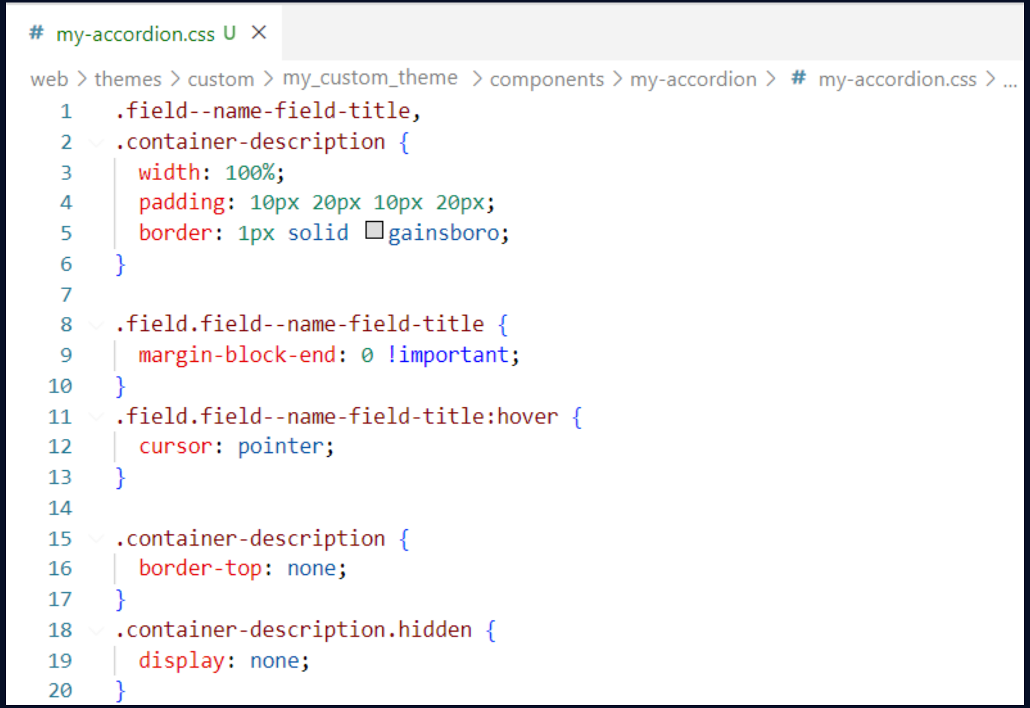
```twig
<div{{ attributes.addClass(classes) }}>
  {{ content|without('field_description') }}
  <div class="container-description hidden">
    {{ content.field_description }}
  </div>
</div>
```



my-accordion.js U ×

web > themes > custom > my_custom_theme > components > my-accordion > my-accordion.js > ...

```javascript
(function ($, Drupal) {
  'use strict';

  Drupal.behaviors.accordion = {
    attach: function (context, settings) {
      $('.paragraph--type--accordion .field--name-field-title').on( "click", function() {
        $(this).siblings('.container-description').toggleClass('hidden');
      } );
    }
  }
})(jQuery, Drupal);
```



my-accordion.css U ×

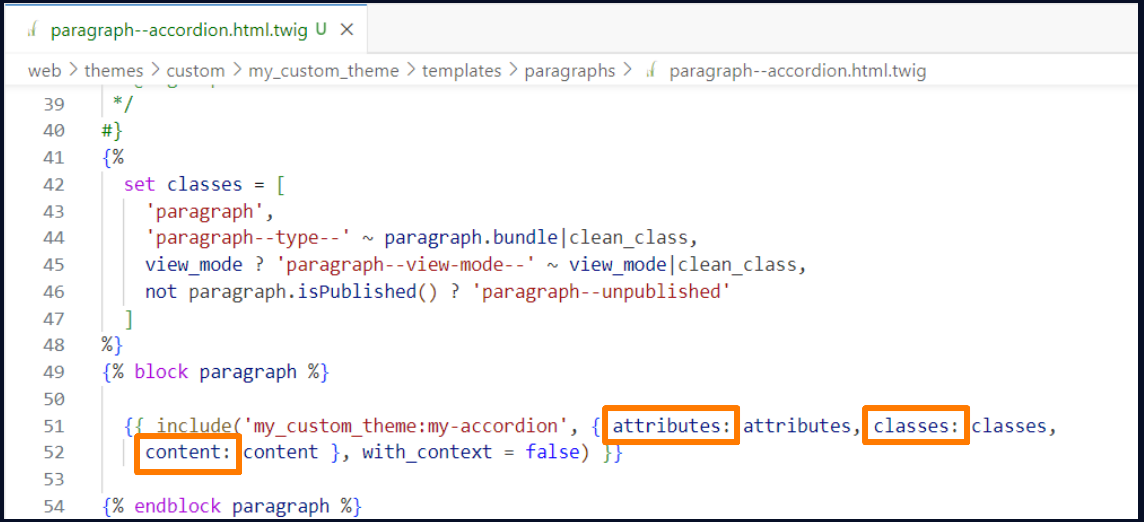
web > themes > custom > my_custom_theme > components > my-accordion > my-accordion.css > ...

```css
.field--name-field-title,
.container-description {
  width: 100%;
  padding: 10px 20px 10px 20px;
  border: 1px solid gainsboro;
}

.field.field--name-field-title {
  margin-block-end: 0 !important;
}
.field.field--name-field-title:hover {
  cursor: pointer;
}

.container-description {
  border-top: none;
}
.container-description.hidden {
  display: none;
}
```

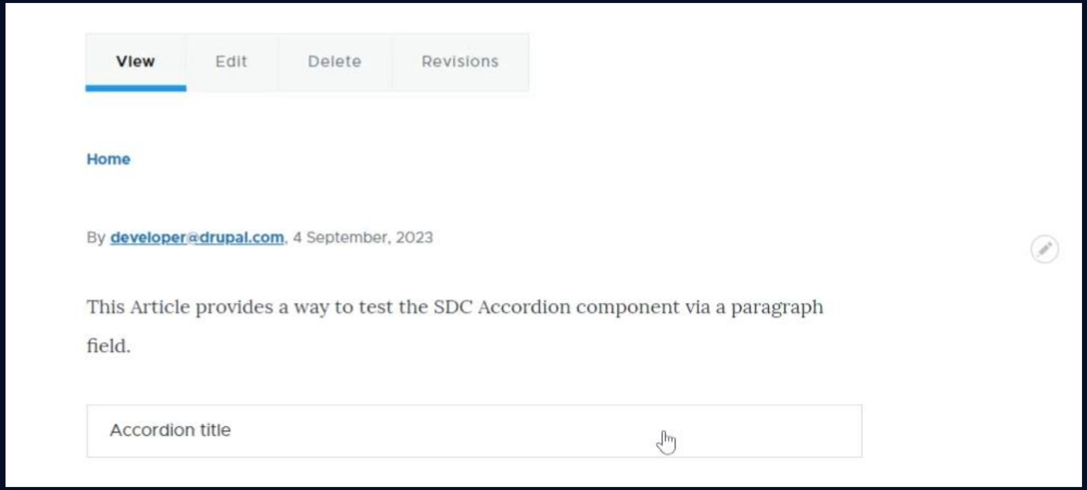
SDC: Example implementation

Include the component

- Within paragraph--accordion.html.twig, reference the newly created component, passing it the desired and/or necessary variables
- Clear the cache!
- Verify the results in an instance of “Article”


paragraph--accordion.html.twig U
web > themes > custom > my_custom_theme > templates > paragraphs > paragraph--accordion.html.twig

```
  */
#}
{%
  set classes = [
    'paragraph',
    'paragraph--type--' ~ paragraph.bundle|clean_class,
    view_mode ? 'paragraph--view-mode--' ~ view_mode|clean_class,
    not paragraph.isPublished() ? 'paragraph--unpublished'
  ]
%}
{% block paragraph %}

  {{ include('my_custom_theme:my-accordion', { attributes: attributes, classes: classes,
  content: content }, with_context = false) }}

{% endblock paragraph %}
```



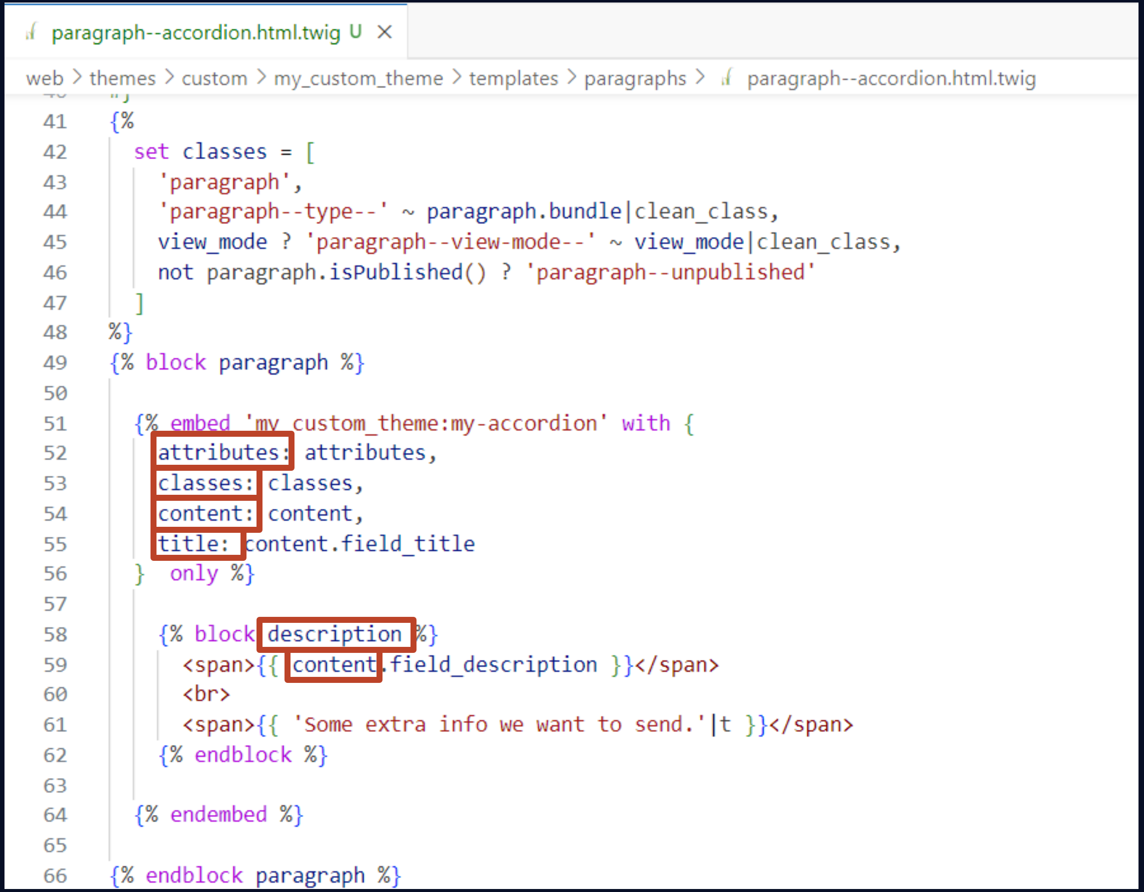
By developer@drupal.com, 4 September, 2023

This Article provides a way to test the SDC Accordion component via a paragraph field.

Accordion title

SDC: Example implementation using slots

paragraph--accordion.html.twig:

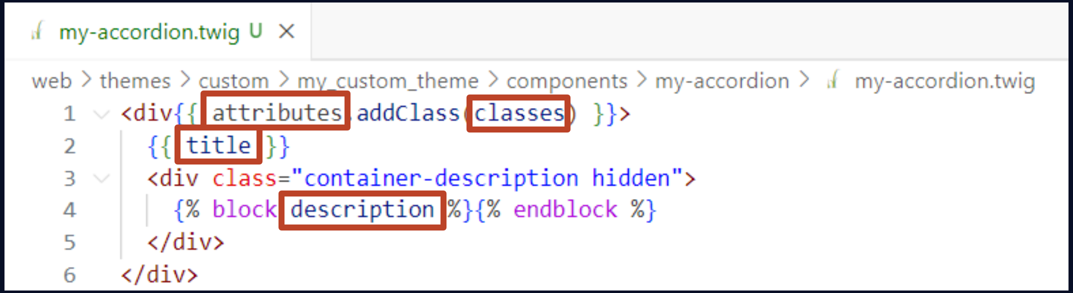


paragraph--accordion.html.twig U ×

web > themes > custom > my_custom_theme > templates > paragraphs > paragraph--accordion.html.twig

```
{%
  set classes = [
    'paragraph',
    'paragraph--type--' ~ paragraph.bundle|clean_class,
    view_mode ? 'paragraph--view-mode--' ~ view_mode|clean_class,
    not paragraph.isPublished() ? 'paragraph--unpublished'
  ]
%}
{% block paragraph %}

  {% embed 'my_custom_theme:my-accordion' with {
    attributes: attributes,
    classes: classes,
    content: content,
    title: content.field_title
  }  only %}

    {% block description %}
      <span>{{ content.field_description }}</span>
      <br>
      <span>{{ 'Some extra info we want to send.'|t }}</span>
    {% endblock %}

  {% endembed %}

{% endblock paragraph %}
```

my-accordion.twig:



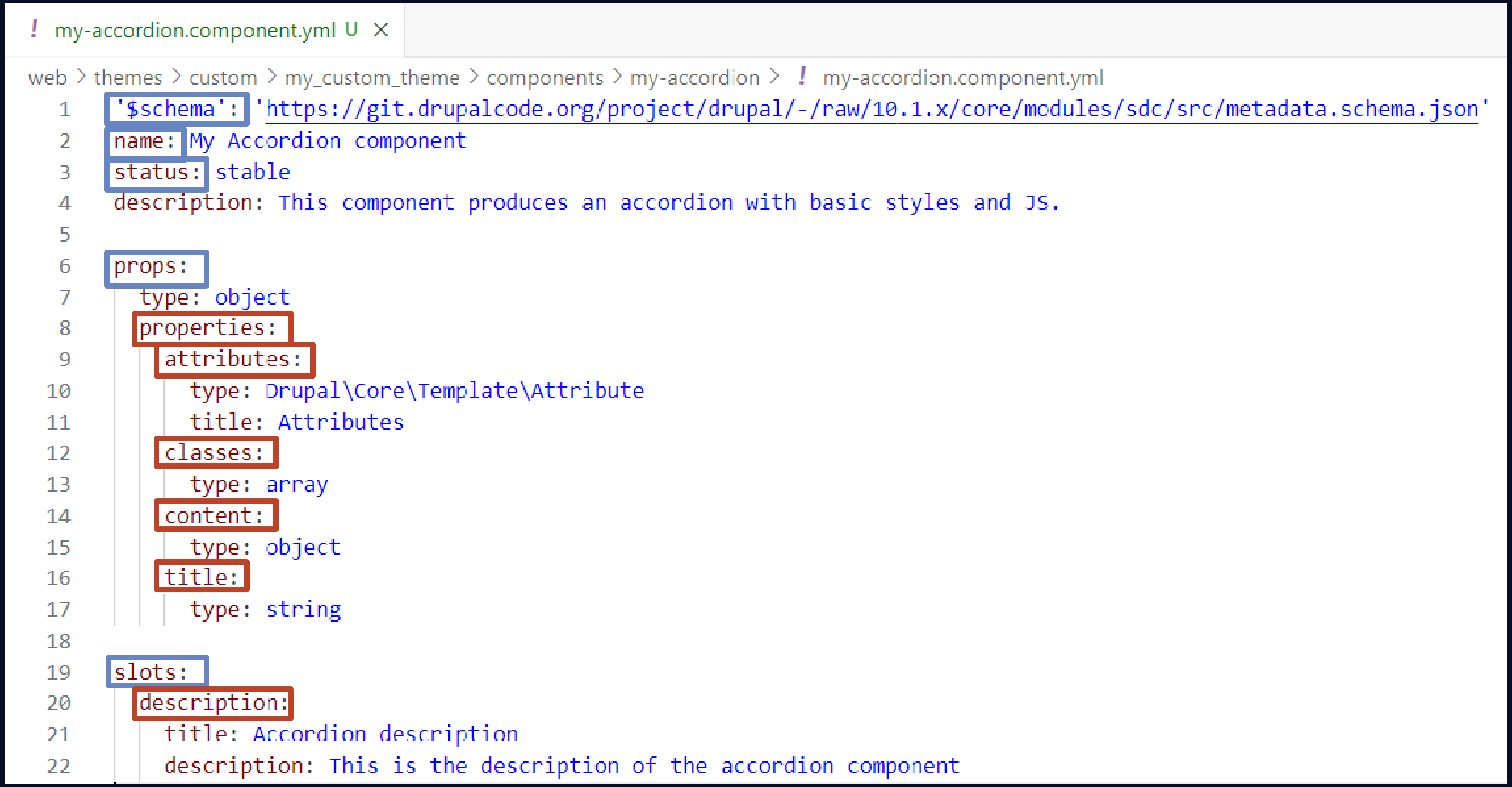
my-accordion.twig U ×

web > themes > custom > my_custom_theme > components > my-accordion > my-accordion.twig

```
<div{{ attributes.addClass(classes) }}>
  {{ title }}
  <div class="container-description hidden">
    {% block description %}{% endblock %}
  </div>
</div>
```

SDC: Example implementation using slots

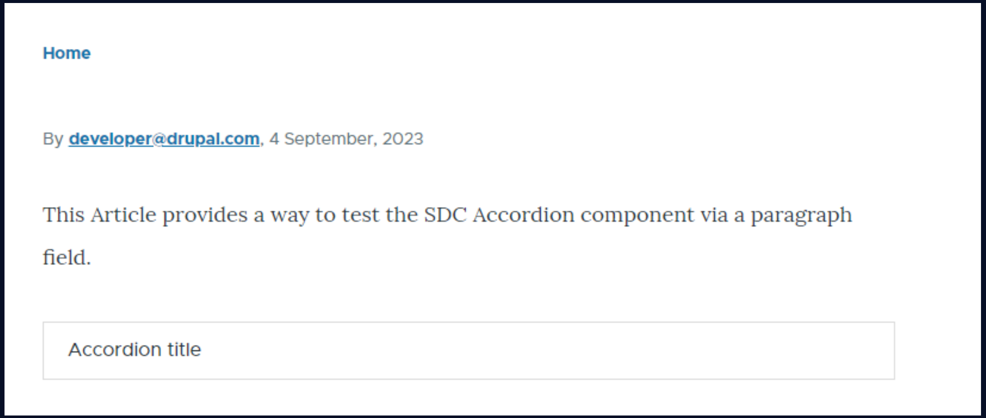
my-accordion.component.yml:


! my-accordion.component.yml U ×
web > themes > custom > my_custom_theme > components > my-accordion > ! my-accordion.component.yml

```yaml
'$schema': 'https://git.drupalcode.org/project/drupal/-/raw/10.1.x/core/modules/sdc/src/metadata.schema.json'
name: My Accordion component
status: stable
description: This component produces an accordion with basic styles and JS.

props:
  type: object
  properties:
    attributes:
      type: Drupal\Core\Template\Attribute
      title: Attributes
    classes:
      type: array
    content:
      type: object
    title:
      type: string

slots:
  description:
    title: Accordion description
    description: This is the description of the accordion component
```

SDC: Example implementation using slots

Results:

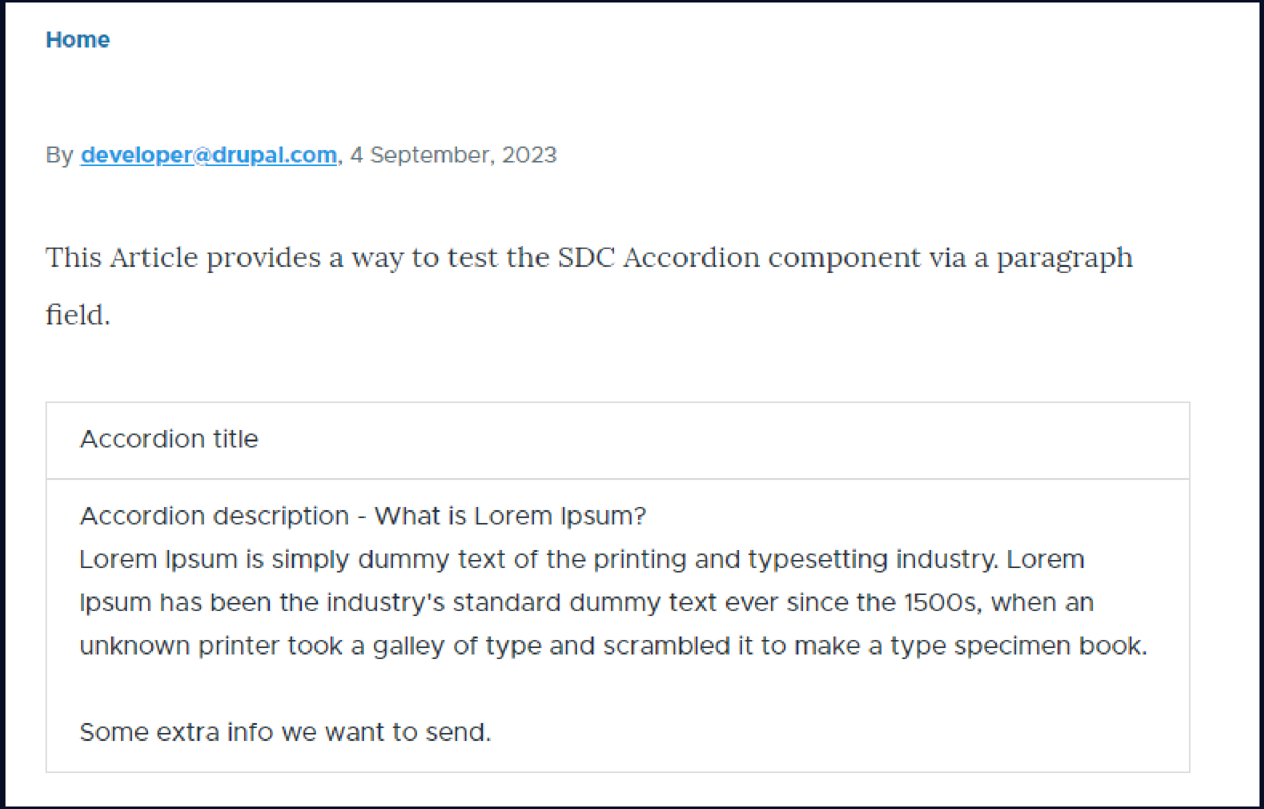


Home

By developer@drupal.com, 4 September, 2023

This Article provides a way to test the SDC Accordion component via a paragraph field.

Accordion title



Home

By developer@drupal.com, 4 September, 2023

This Article provides a way to test the SDC Accordion component via a paragraph field.

Accordion title

Accordion description - What is Lorem Ipsum?
Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.

Some extra info we want to send.

SDC: Example implementation

Overall method

- Create the /components directory
- Create a directory for your component
- Include a .component.yml file
- Include whatever markup, styling and js files you might need
- Include your new component within a Drupal template (otherwise it does nothing)

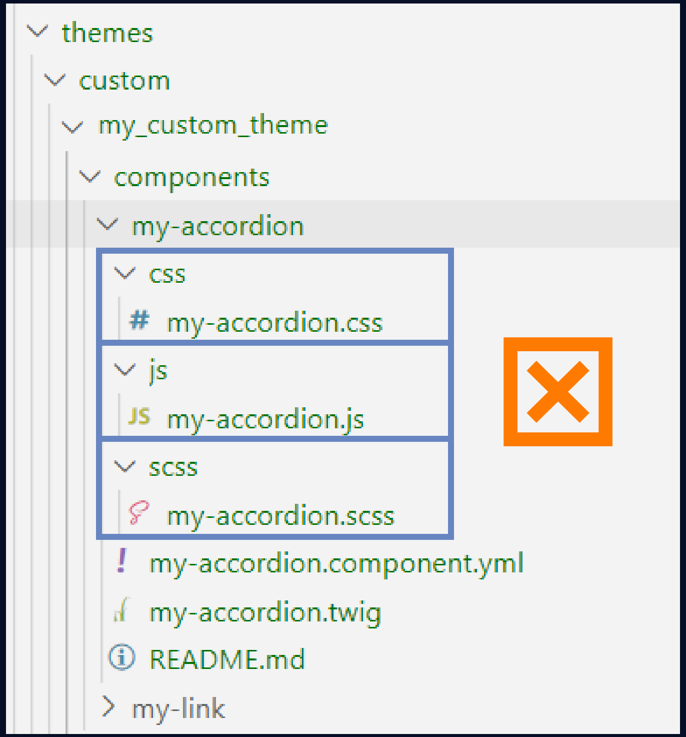
03

Common mistakes and best practices

- Directory management
- Compiling the CSS
- Calling the component
- README file

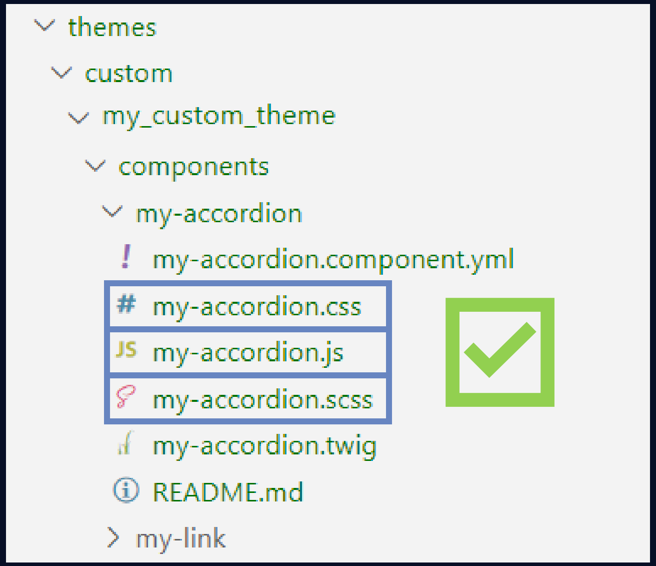
Common mistakes: Directory management

Directories for each type of file



```
themes
  custom
    my_custom_theme
      components
        my-accordion
          css
            my-accordion.css
          js
            my-accordion.js
          scss
            my-accordion.scss
          my-accordion.component.yml
          my-accordion.twig
          README.md
        my-link
```

All styling files within the component's main directory

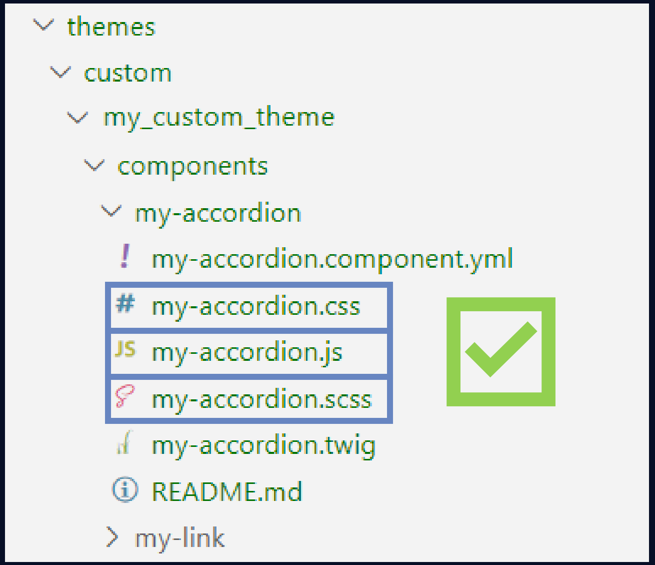


```
themes
  custom
    my_custom_theme
      components
        my-accordion
          my-accordion.component.yml
          my-accordion.css
          my-accordion.js
          my-accordion.scss
          my-accordion.twig
          README.md
        my-link
```

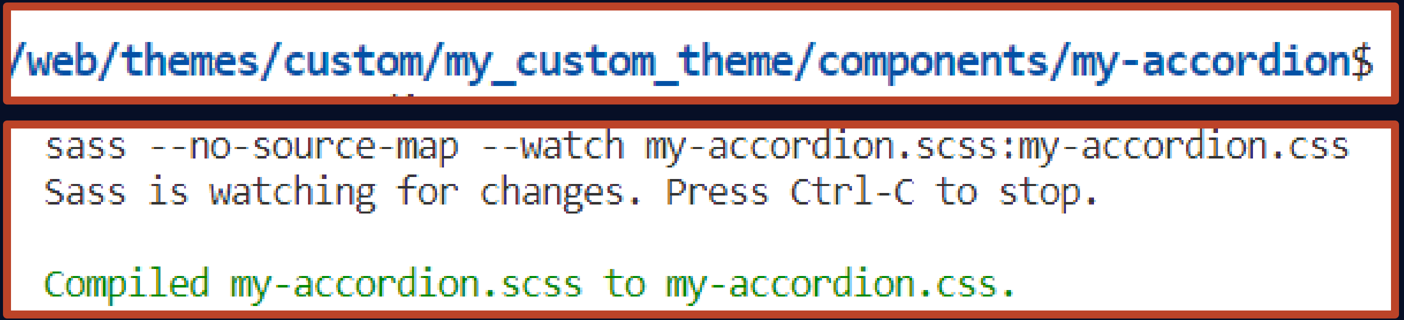
Common mistakes: Compiling the CSS

All styling files within the component's main directory

Because all files are placed within the same directory, remember that the CSS is now compiled within your component's directory!



```
themes
  custom
    my_custom_theme
      components
        my-accordion
          my-accordion.component.yml
          my-accordion.css
          my-accordion.js
          my-accordion.scss
          my-accordion.twig
          README.md
        my-link
```



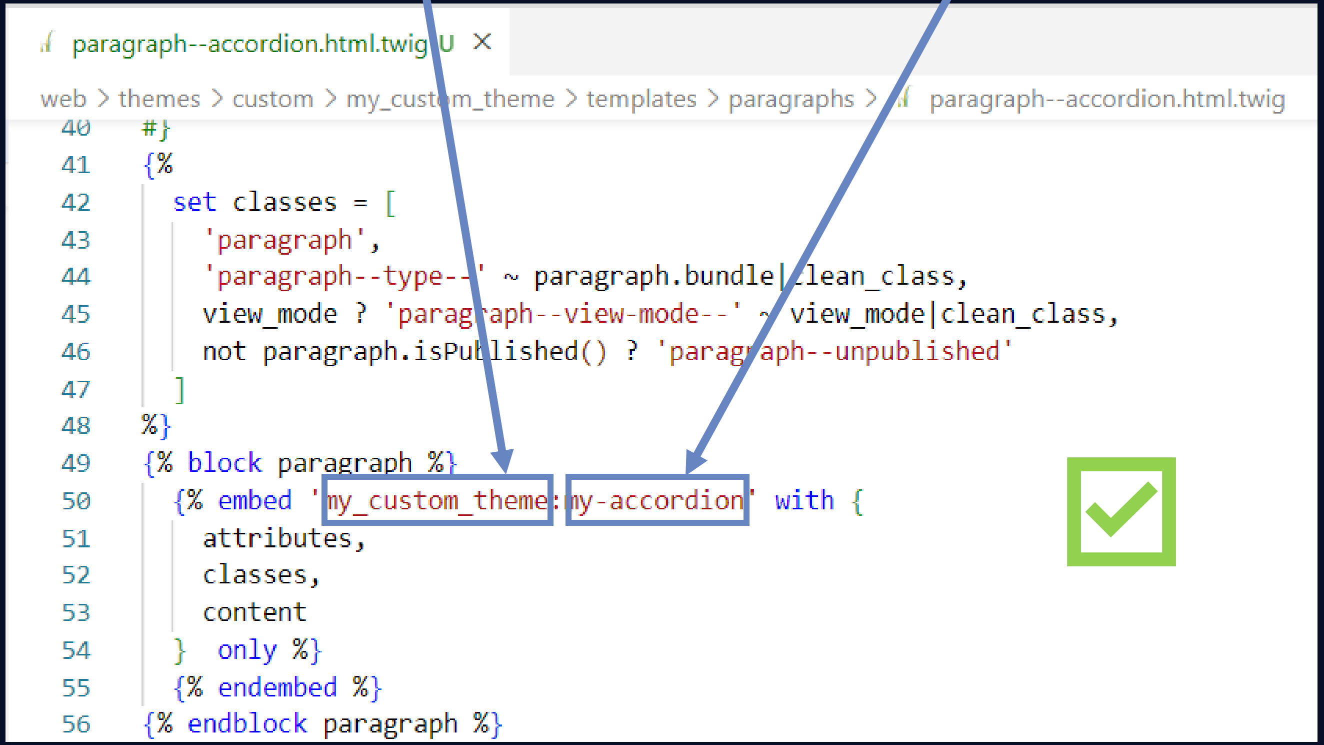
```
/web/themes/custom/my_custom_theme/components/my-accordion$
```

```
sass --no-source-map --watch my-accordion.scss:my-accordion.css
Sass is watching for changes. Press Ctrl-C to stop.

Compiled my-accordion.scss to my-accordion.css.
```

Common mistakes: Calling the component

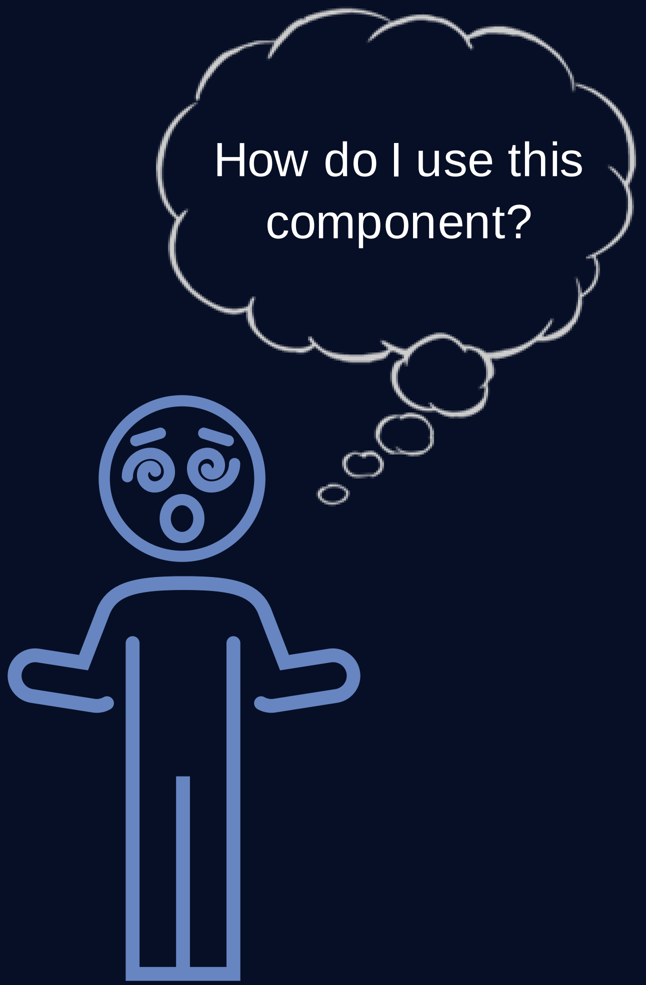
Your theme or module's machine name

Your component's machine name

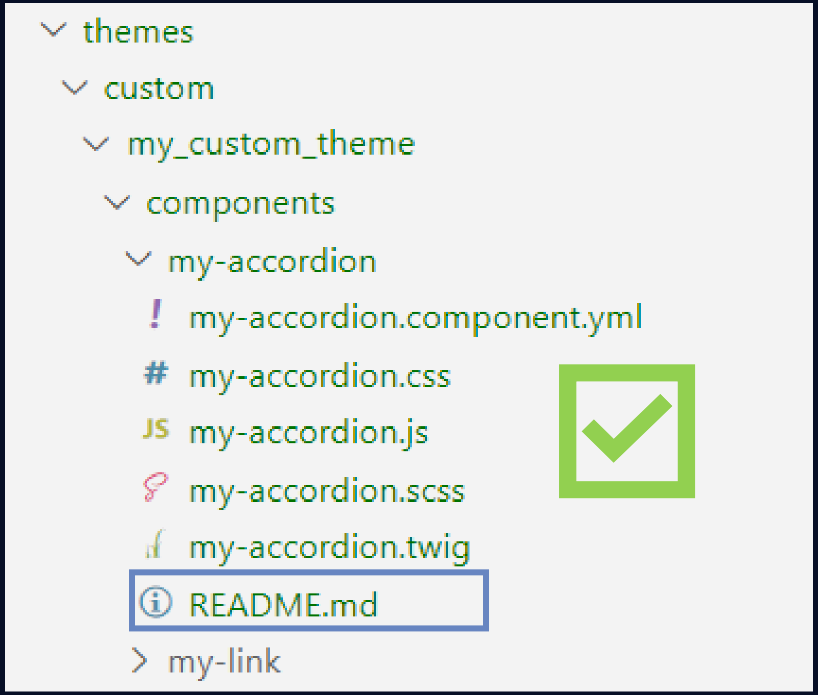

paragraph--accordion.html.twig U
web > themes > custom > my_custom_theme > templates > paragraphs > paragraph--accordion.html.twig

```
#}
{%
  set classes = [
    'paragraph',
    'paragraph--type--' ~ paragraph.bundle|clean_class,
    view_mode ? 'paragraph--view-mode--' ~ view_mode|clean_class,
    not paragraph.isPublished() ? 'paragraph--unpublished'
  ]
%}
{% block paragraph %}
  {% embed 'my_custom_theme:my-accordion' with {
    attributes,
    classes,
    content
  }  only %}
  {% endembed %}
{% endblock paragraph %}
```

Best practices: README file



Documentation, documentation, documentation!



```
themes
  custom
    my_custom_theme
      components
        my-accordion
          my-accordion.component.yml
          my-accordion.css
          my-accordion.js
          my-accordion.scss
          my-accordion.twig
          README.md
        my-link
```

04

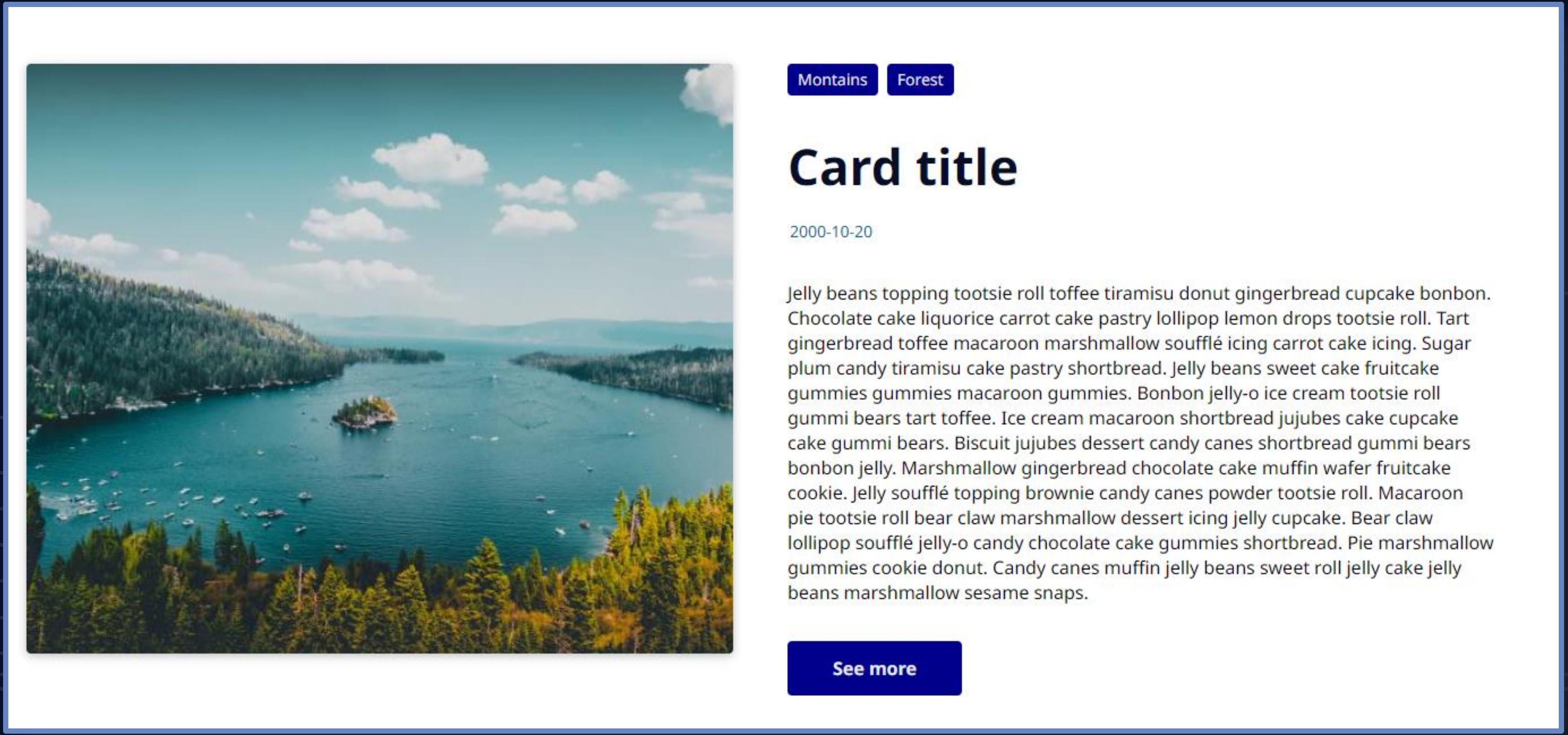
The Power of Integration: SDC with Complementary Modules

- Exploring How Other Drupal Modules Can Complement the Functionality of SDC

SDC with Complementary Modules

How can we see the components made in Single Direct Components, and choose what to use?

Imagine that you create several components for your Drupal website using SDC,
and you want to choose one component for a specific field/paragraph. How can you do this?



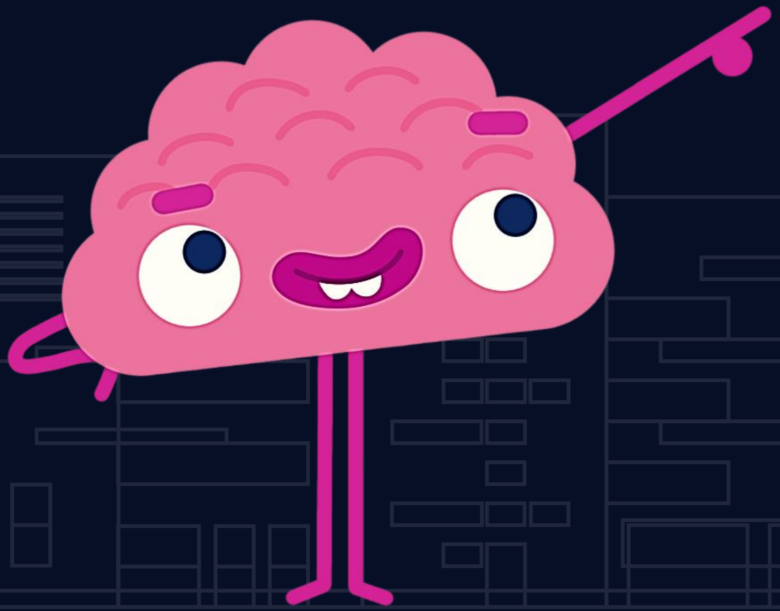
Card title

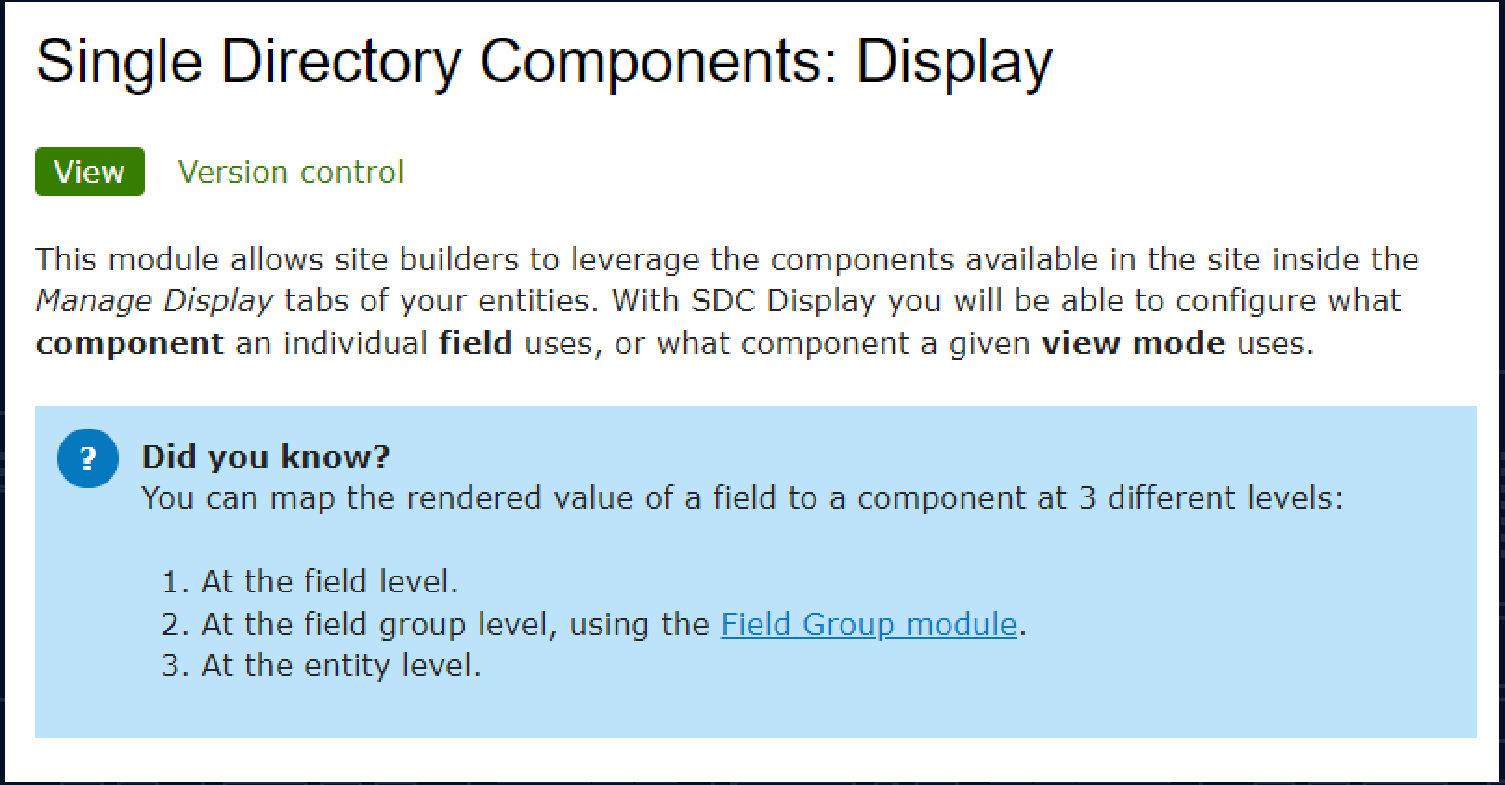
2000-10-20

Jelly beans topping tootsie roll toffee tiramisu donut gingerbread cupcake bonbon. Chocolate cake liquorice carrot cake pastry lollipop lemon drops tootsie roll. Tart gingerbread toffee macaroon marshmallow soufflé icing carrot cake icing. Sugar plum candy tiramisu cake pastry shortbread. Jelly beans sweet cake fruitcake gummies gummies macaroon gummies. Bonbon jelly-o ice cream tootsie roll gummi bears tart toffee. Ice cream macaroon shortbread jujubes cake cupcake cake gummi bears. Biscuit jujubes dessert candy canes shortbread gummi bears bonbon jelly. Marshmallow gingerbread chocolate cake muffin wafer fruitcake cookie. Jelly soufflé topping brownie candy canes powder tootsie roll. Macaroon pie tootsie roll bear claw marshmallow dessert icing jelly cupcake. Bear claw lollipop soufflé jelly-o candy chocolate cake gummies shortbread. Pie marshmallow gummies cookie donut. Candy canes muffin jelly beans sweet roll jelly cake jelly beans marshmallow sesame snaps.

SDC with Complementary Modules

Mateu Aguiló Bosch (e0ipso)





Single Directory Components: Display

This module allows site builders to leverage the components available in the site inside the *Manage Display* tabs of your entities. With SDC Display you will be able to configure what **component** an individual **field** uses, or what component a given **view mode** uses.

Did you know?
You can map the rendered value of a field to a component at 3 different levels:

1. At the field level.
2. At the field group level, using the Field Group module.
3. At the entity level.

https://www.drupal.org/project/sdc_display
https://video.mateuaguilo.com/w/sC5nv52GLQrPHYcjBUvEeN

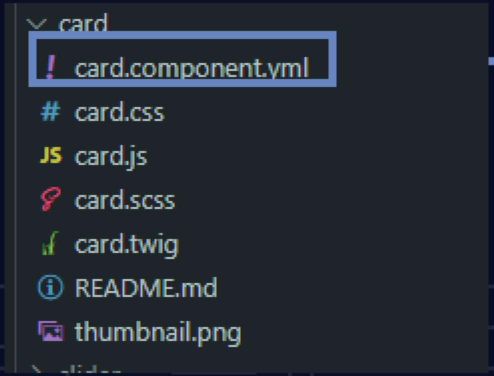
SDC with Complementary Modules

How can we use sdc_display?

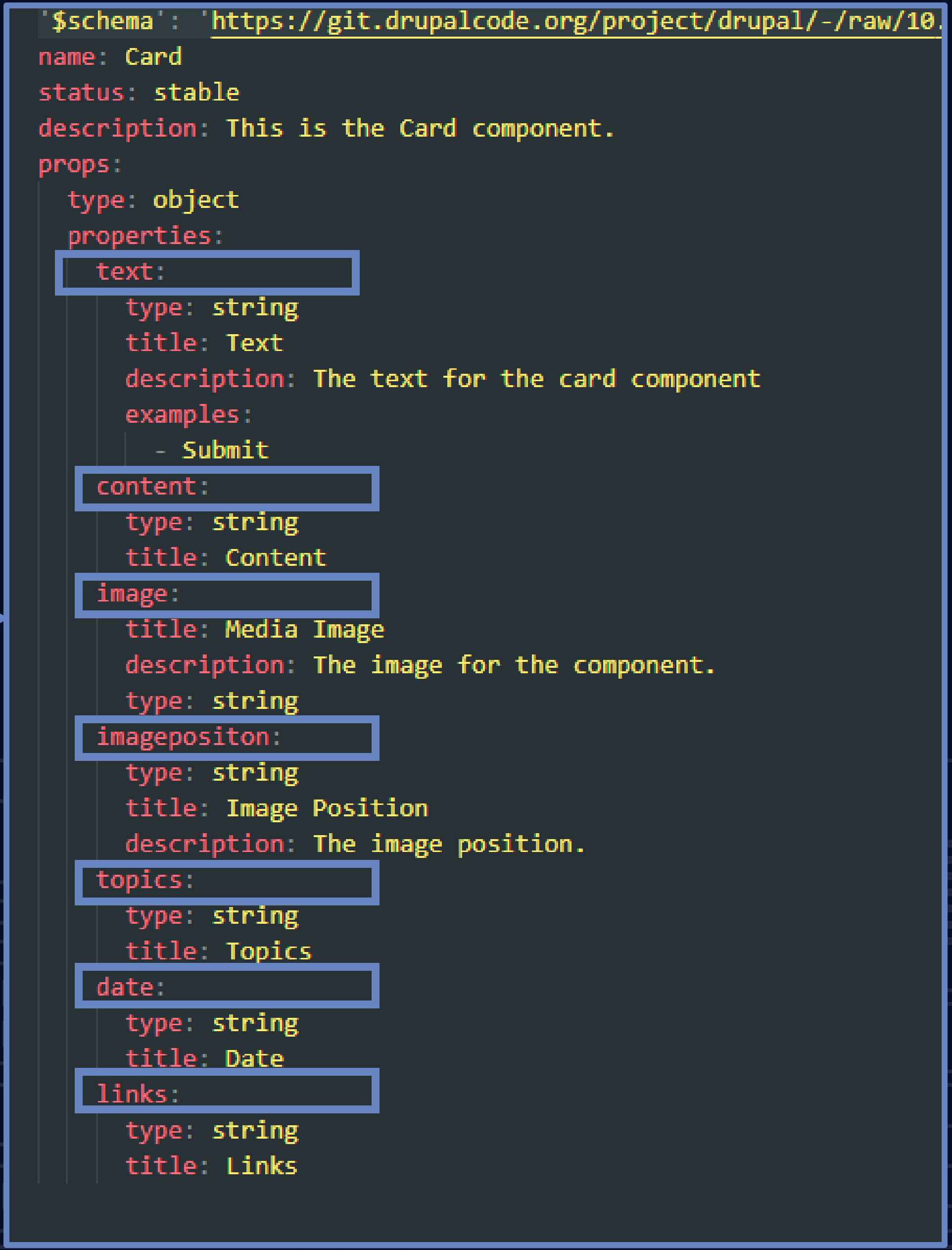
After install, the module.

We need to create the fields that our component need.

To see the fields we go to the file **card.component.yml**



```
card
  card.component.yml
  card.css
  card.js
  card.scss
  card.twig
  README.md
  thumbnail.png
```



```yaml
'$schema': 'https://git.drupalcode.org/project/drupal/-/raw/10.
name: Card
status: stable
description: This is the Card component.
props:
  type: object
  properties:
    text:
      type: string
      title: Text
      description: The text for the card component
      examples:
        - Submit
    content:
      type: string
      title: Content
    image:
      title: Media Image
      description: The image for the component.
      type: string
    imagepositon:
      type: string
      title: Image Position
      description: The image position.
    topics:
      type: string
      title: Topics
    date:
      type: string
      title: Date
    links:
      type: string
      title: Links
```

After create the fields we need to make a question.

We know that the **component card** is a component, but inside of this component.

What these fields can be a component too?

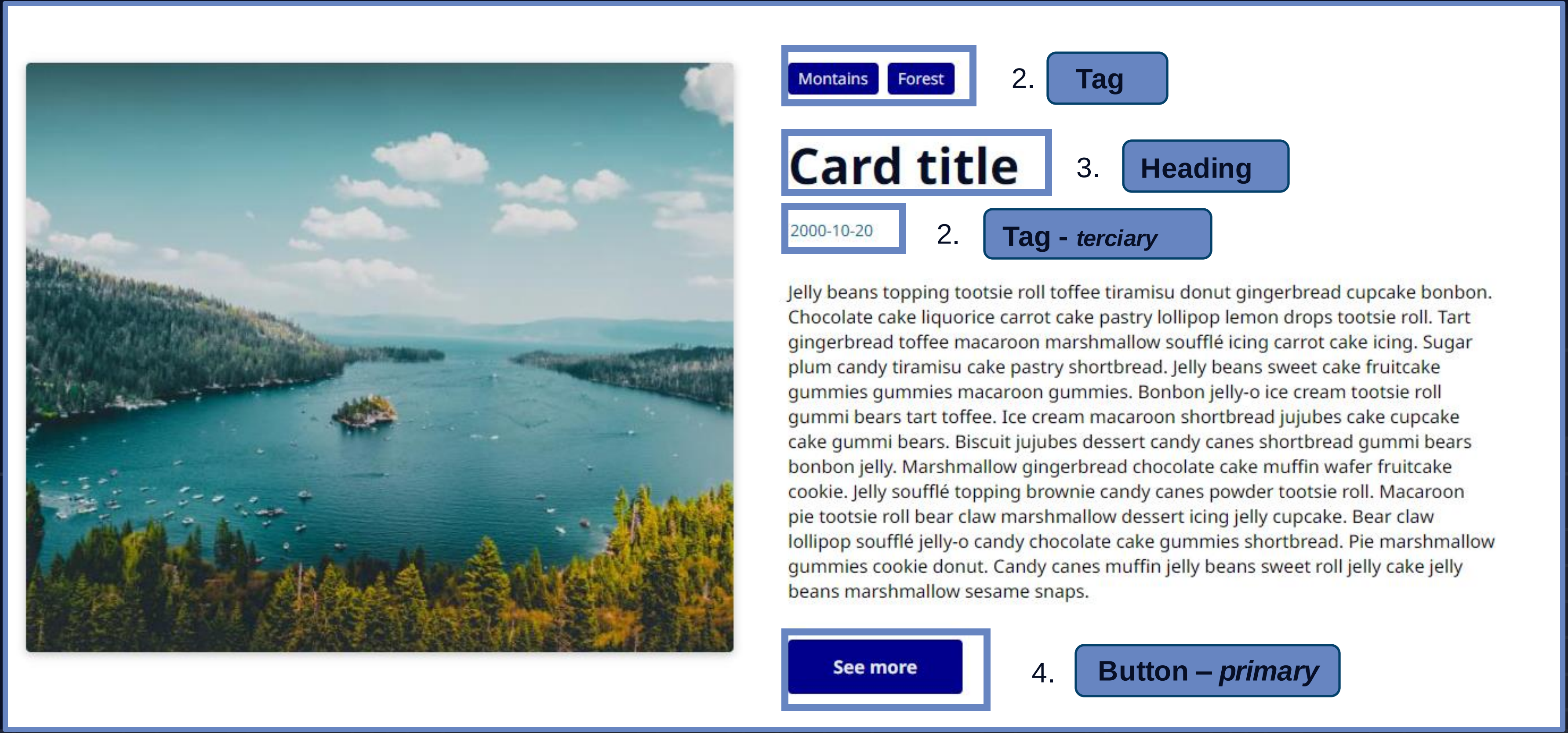
SDC with Complementary Modules

What is a component?

A component is an element that we want to reuse throughout the site.

1. Card
2. Tag
3. Heading
4. Button

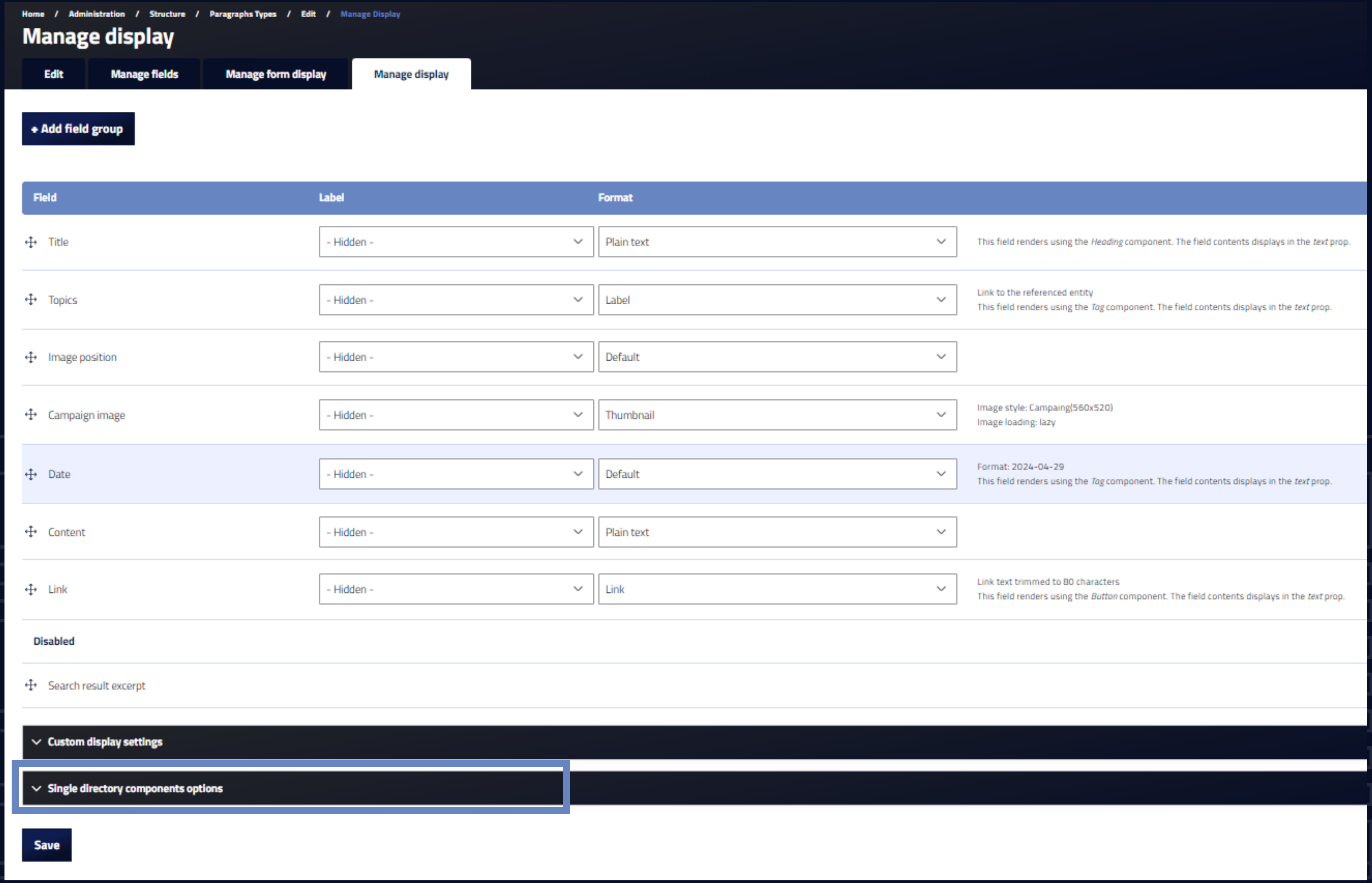
1.



Jelly beans topping tootsie roll toffee tiramisu donut gingerbread cupcake bonbon. Chocolate cake liquorice carrot cake pastry lollipop lemon drops tootsie roll. Tart gingerbread toffee macaroon marshmallow soufflé icing carrot cake icing. Sugar plum candy tiramisu cake pastry shortbread. Jelly beans sweet cake fruitcake gummies gummies macaroon gummies. Bonbon jelly-o ice cream tootsie roll gummi bears tart toffee. Ice cream macaroon shortbread jujubes cake cupcake cake gummi bears. Biscuit jujubes dessert candy canes shortbread gummi bears bonbon jelly. Marshmallow gingerbread chocolate cake muffin wafer fruitcake cookie. Jelly soufflé topping brownie candy canes powder tootsie roll. Macaroon pie tootsie roll bear claw marshmallow dessert icing jelly cupcake. Bear claw lollipop soufflé jelly-o candy chocolate cake gummies shortbread. Pie marshmallow gummies cookie donut. Candy canes muffin jelly beans sweet roll jelly cake jelly beans marshmallow sesame snaps.

SDC with Complementary Modules

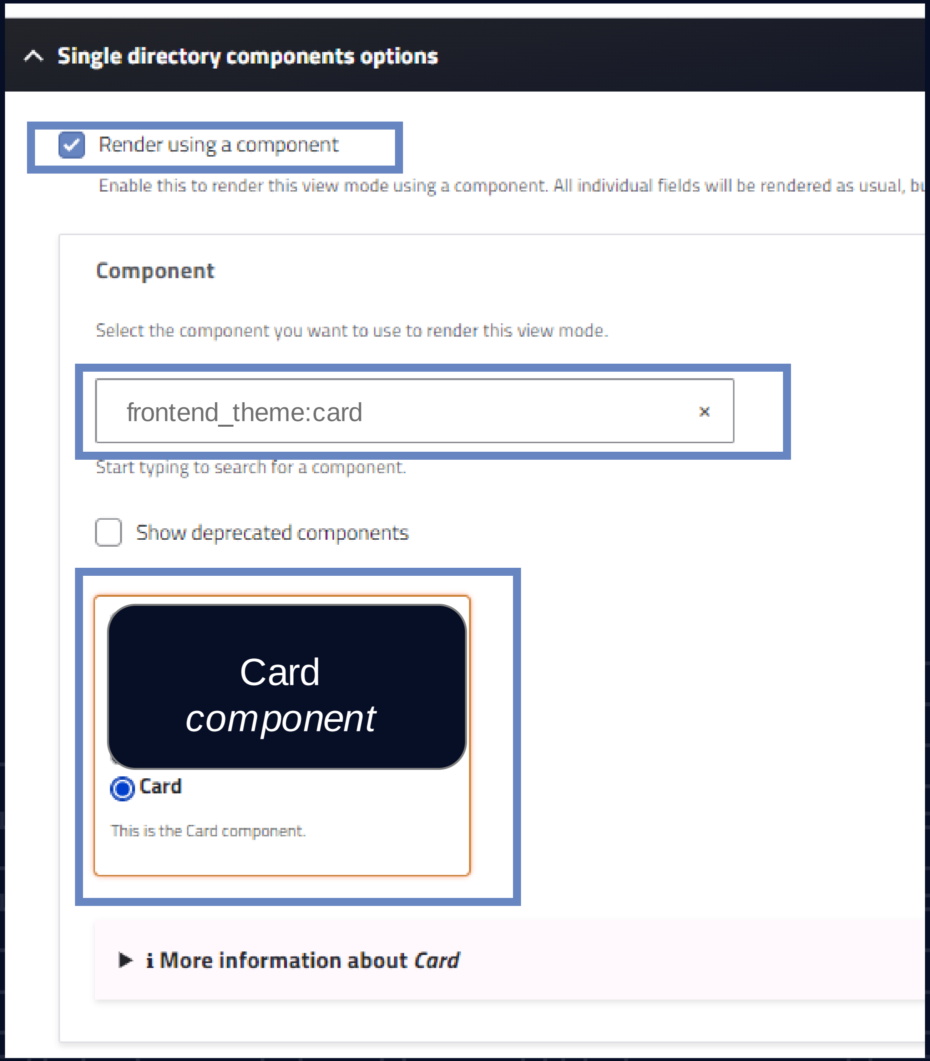
Choose component and mapping the fields on manage display


Home / Administration / Structure / Paragraphs Types / Edit / Manage Display
Edit
Manage fields
Manage form display
Manage display
+ Add field group

Manage display

Field	Label	Format	
Title	- Hidden -	Plain text	This field renders using the *Heading* component. The field contents displays in the *text* prop.
Topics	- Hidden -	Label	Link to the referenced entity This field renders using the *Tag* component. The field contents displays in the *text* prop.
Image position	- Hidden -	Default	
Campaign image	- Hidden -	Thumbnail	Image style: Campaing(560x520) Image loading: lazy
Date	- Hidden -	Default	Format: 2024-04-29 This field renders using the *Tag* component. The field contents displays in the *text* prop.
Content	- Hidden -	Plain text	
Link	- Hidden -	Link	Link text trimmed to 80 characters This field renders using the *Button* component. The field contents displays in the *text* prop.
Disabled			
Search result excerpt			

SDC with Complementary Modules



Single directory components options

[✓] Render using a component

Enable this to render this view mode using a component. All individual fields will be rendered as usual, bu

Component

Select the component you want to use to render this view mode.

frontend_theme:card ×

Start typing to search for a component.

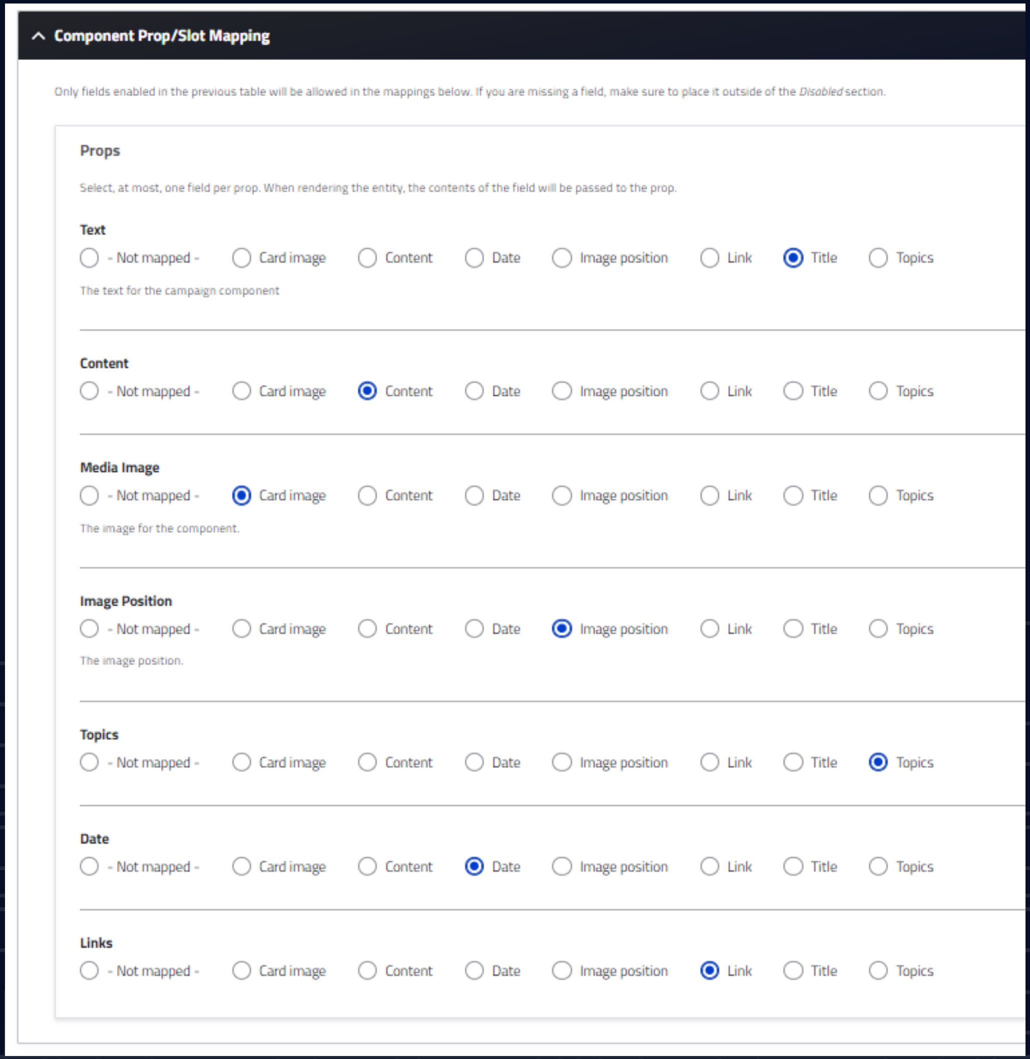
[ ] Show deprecated components

Card
component

[● Card]

This is the Card component.

▶ i More information about *Card*



Component Prop/Slot Mapping

Only fields enabled in the previous table will be allowed in the mappings below. If you are missing a field, make sure to place it outside of the *Disabled* section.

Props

Select, at most, one field per prop. When rendering the entity, the contents of the field will be passed to the prop.

Text
- Not mapped - · Card image · Content · Date · Image position · Link · [✓ Title] · Topics
The text for the campaign component

Content
- Not mapped - · Card image · [✓ Content] · Date · Image position · Link · Title · Topics

Media Image
- Not mapped - · [✓ Card image] · Content · Date · Image position · Link · Title · Topics
The image for the component.

Image Position
- Not mapped - · Card image · Content · Date · [✓ Image position] · Link · Title · Topics
The image position.

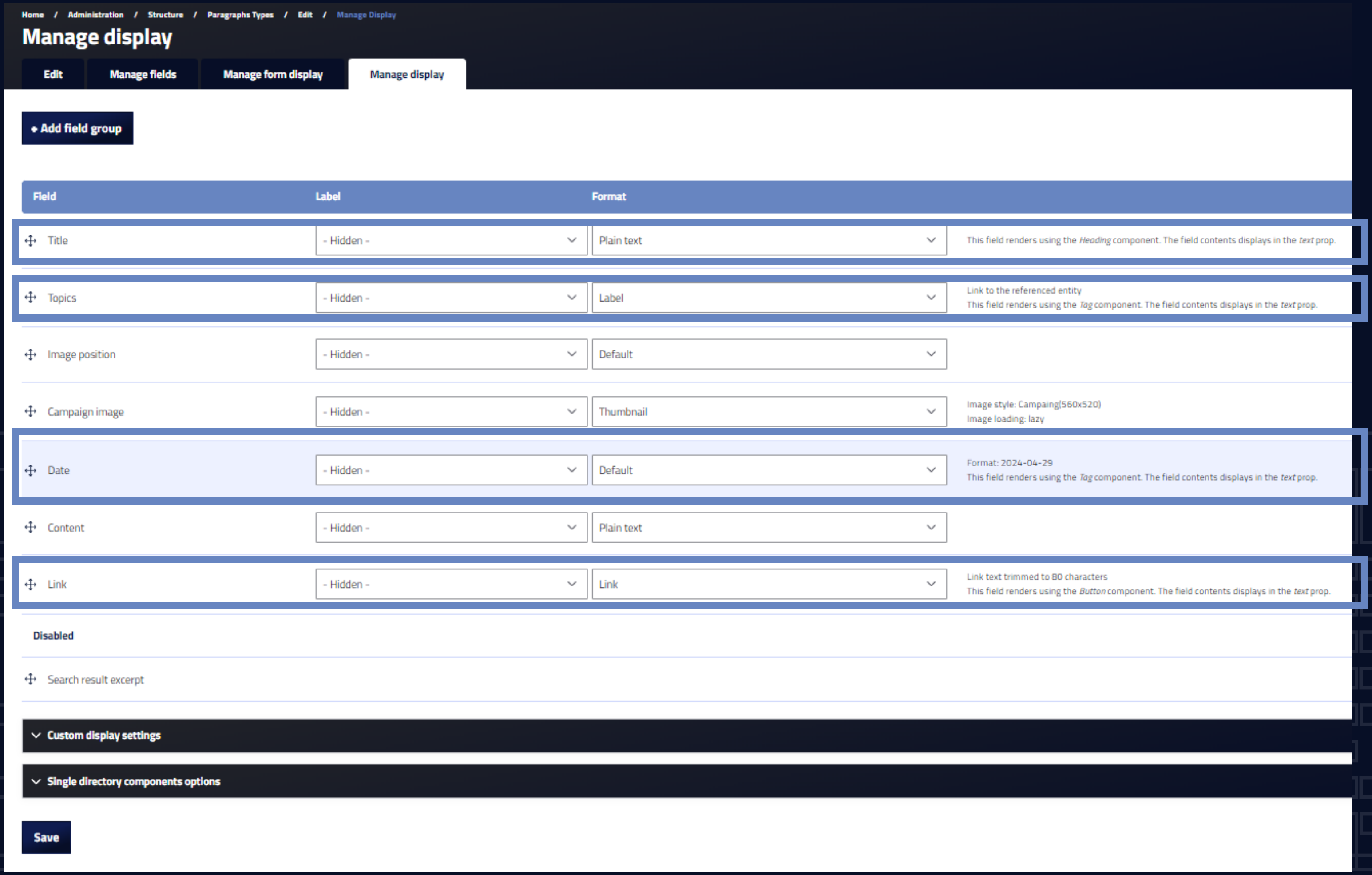
Topics
- Not mapped - · Card image · Content · Date · Image position · Link · Title · [✓ Topics]

Date
- Not mapped - · Card image · Content · [✓ Date] · Image position · Link · Title · Topics

Links
- Not mapped - · Card image · Content · Date · Image position · [✓ Link] · Title · Topics

SDC with Complementary Modules

Render field as a component



Home / Administration / Structure / Paragraphs Types / Edit / Manage Display

Manage display

Edit | Manage fields | Manage form display | Manage display

+ Add field group

Field	Label	Format	
Title	- Hidden -	Plain text	This field renders using the *Heading* component. The field contents displays in the *text* prop.
Topics	- Hidden -	Label	Link to the referenced entity This field renders using the *Tag* component. The field contents displays in the *text* prop.
Image position	- Hidden -	Default	
Campaign image	- Hidden -	Thumbnail	Image style: Campaing(560x520) Image loading: lazy
Date	- Hidden -	Default	Format: 2024-04-29 This field renders using the *Tag* component. The field contents displays in the *text* prop.
Content	- Hidden -	Plain text	
Link	- Hidden -	Link	Link text trimmed to 80 characters This field renders using the *Button* component. The field contents displays in the *text* prop.
Disabled			
Search result excerpt			

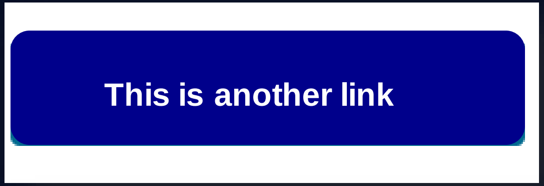
Custom display settings

Single directory components options

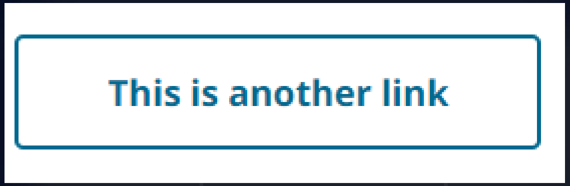
Save

SDC with Complementary Modules

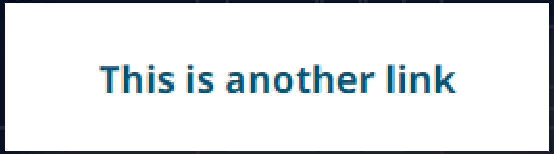
Render field as a component


This is another link

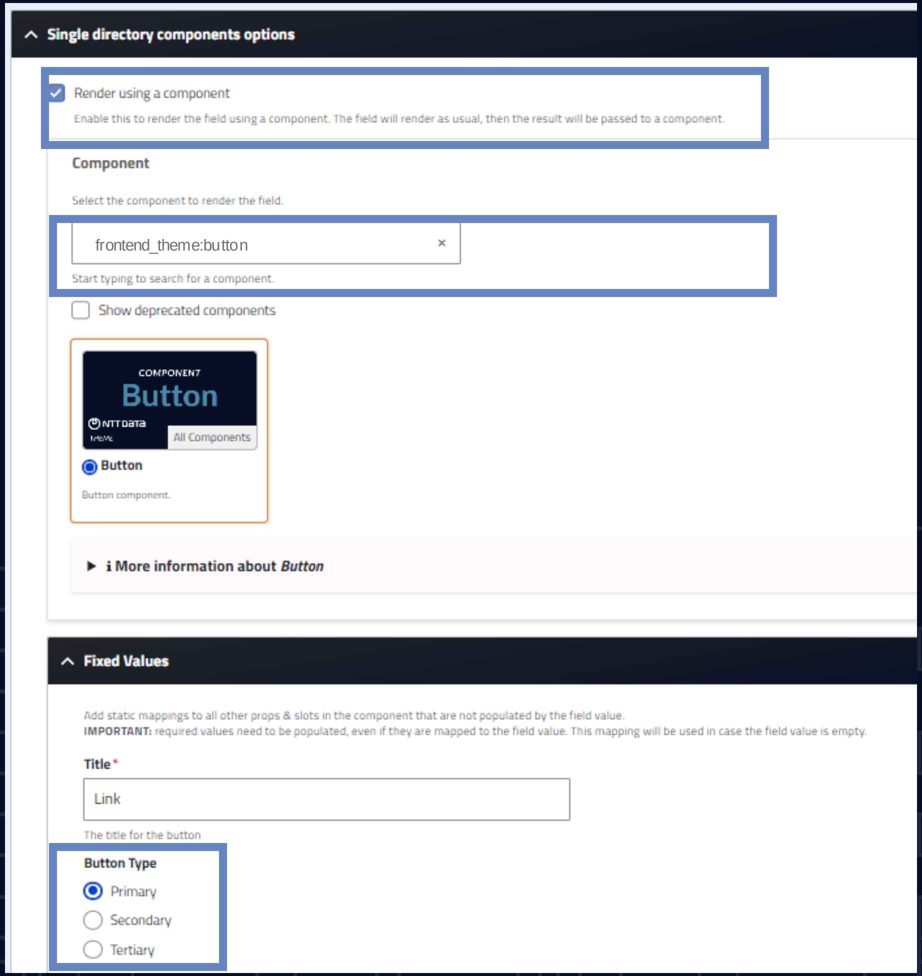
Button - *primary*


This is another link

Button - *secondary*


This is another link

Button - *Terciary*


Single directory components options

[✓] Render using a component

Enable this to render the field using a component. The field will render as usual, then the result will be passed to a component.

Component

Select the component to render the field.

frontend_theme:button ×

Start typing to search for a component.

[ ] Show deprecated components

[✓ Button]

Button component.

▶ i More information about *Button*

Fixed Values

Add static mappings to all other props & slots in the component that are not populated by the field value.
IMPORTANT: required values need to be populated, even if they are mapped to the field value. This mapping will be used in case the field value is empty.

Title *

Link

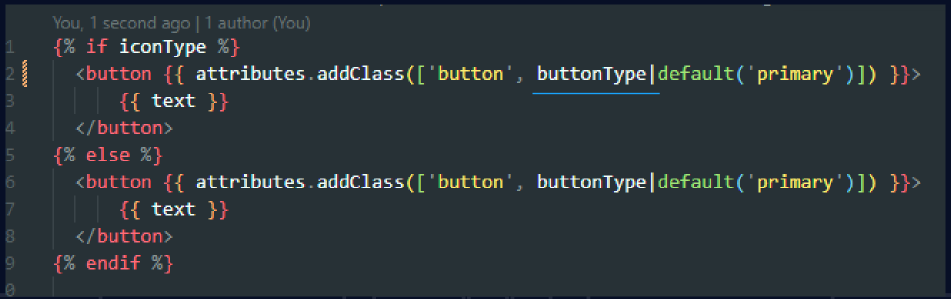
The title for the button

Button Type

[✓ Primary]
Secondary
Tertiary

SDC with Complementary Modules

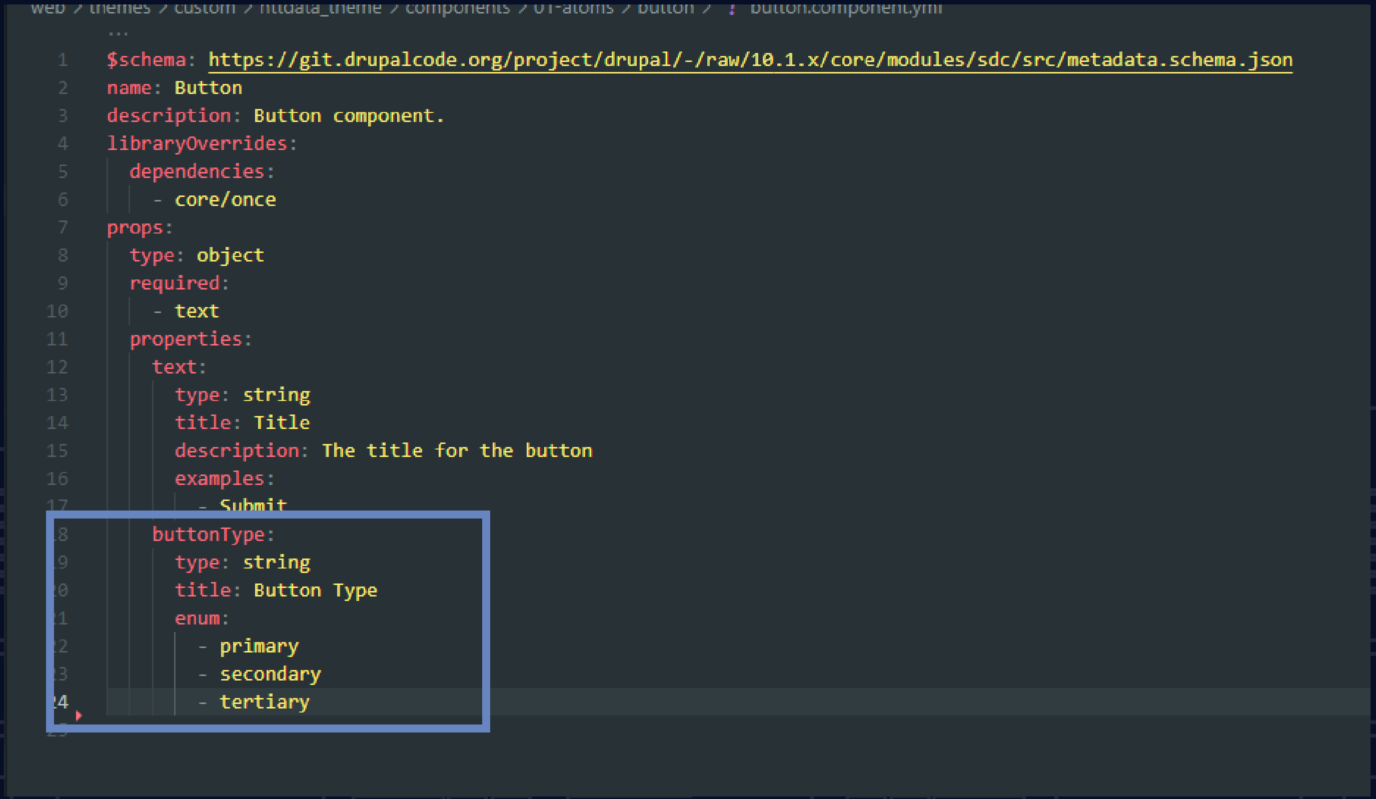
Options for component



You, 1 second ago | 1 author (You)

```twig
{% if iconType %}
  <button {{ attributes.addClass(['button', buttonType|default('primary')]) }}>
      {{ text }}
  </button>
{% else %}
  <button {{ attributes.addClass(['button', buttonType|default('primary')]) }}>
      {{ text }}
  </button>
{% endif %}
```

button.twig



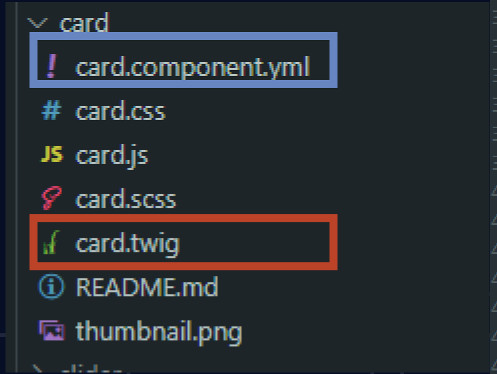
web › themes › custom › nttdata_theme › components › 01-atoms › button › ! button.component.yml

```yaml
$schema: https://git.drupalcode.org/project/drupal/-/raw/10.1.x/core/modules/sdc/src/metadata.schema.json
name: Button
description: Button component.
libraryOverrides:
  dependencies:
    - core/once
props:
  type: object
  required:
    - text
  properties:
    text:
      type: string
      title: Title
      description: The title for the button
      examples:
        - Submit
    buttonType:
      type: string
      title: Button Type
      enum:
        - primary
        - secondary
        - tertiary
```

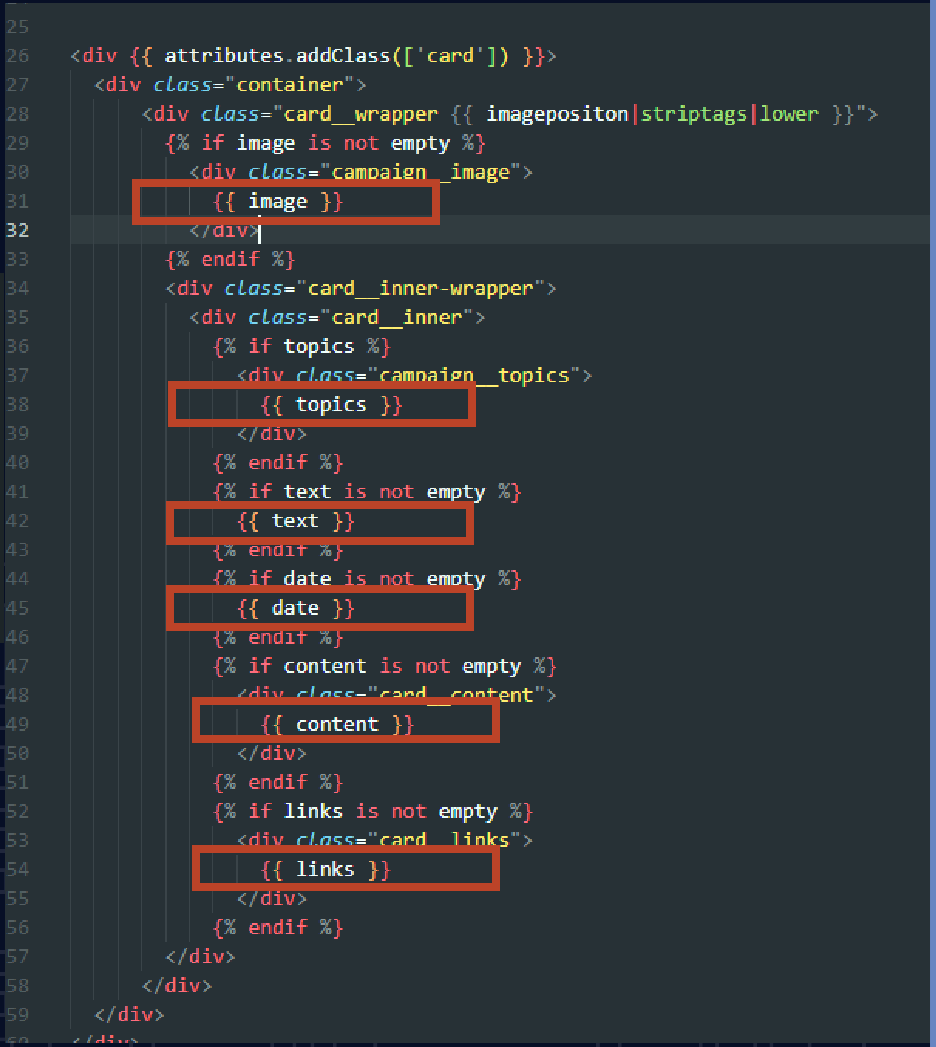
button.component.yml

SDC with Complementary Modules

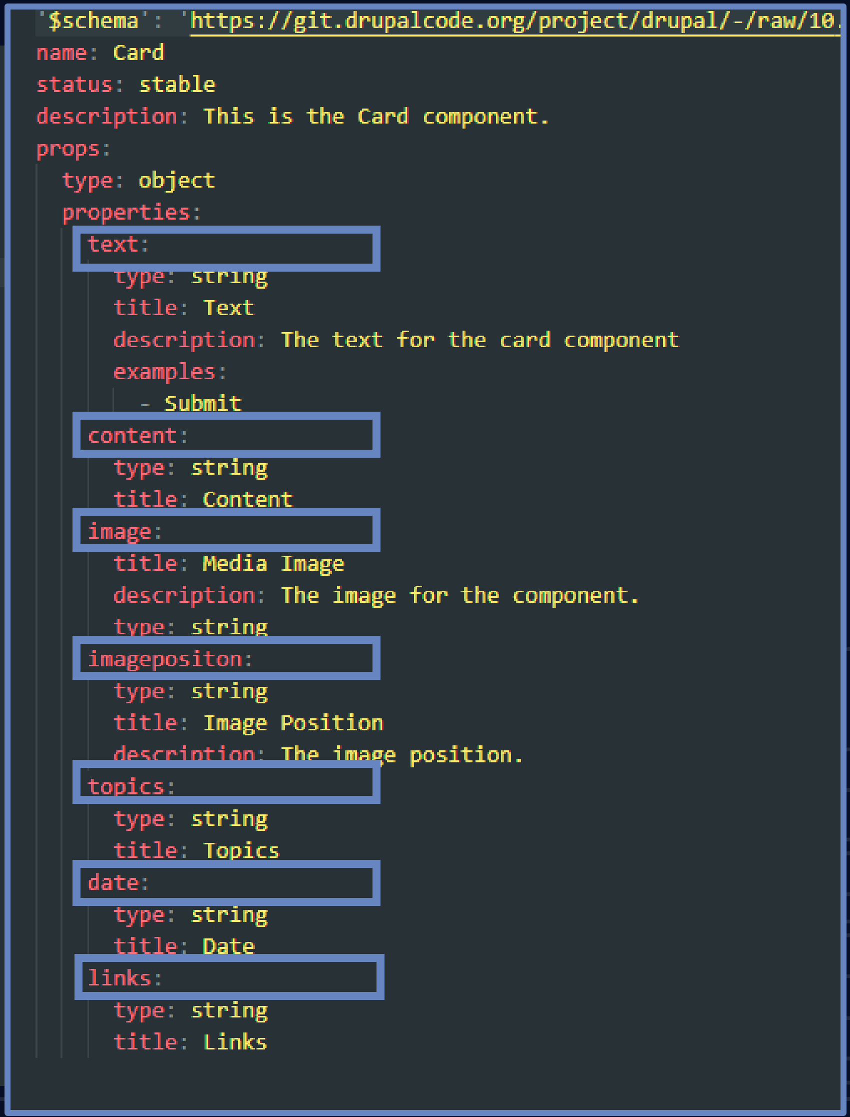
Notes



```
∨ card
  card.component.yml
  card.css
  card.js
  card.scss
  card.twig
  README.md
  thumbnail.png
```



```twig

<div {{ attributes.addClass(['card']) }}>
  <div class="container">
      <div class="card__wrapper {{ imagepositon|striptags|lower }}">
        {% if image is not empty %}
          <div class="campaign__image">
            {{ image }}
          </div>
        {% endif %}
        <div class="card__inner-wrapper">
          <div class="card__inner">
            {% if topics %}
              <div class="campaign__topics">
                {{ topics }}
              </div>
            {% endif %}
            {% if text is not empty %}
              {{ text }}
            {% endif %}
            {% if date is not empty %}
              {{ date }}
            {% endif %}
            {% if content is not empty %}
              <div class="card__content">
                {{ content }}
              </div>
            {% endif %}
            {% if links is not empty %}
              <div class="card__links">
                {{ links }}
              </div>
            {% endif %}
        </div>
      </div>
  </div>
```



```yaml
'$schema': 'https://git.drupalcode.org/project/drupal/-/raw/10
name: Card
status: stable
description: This is the Card component.
props:
  type: object
  properties:
    text:
      type: string
      title: Text
      description: The text for the card component
      examples:
        - Submit
    content:
      type: string
      title: Content
    image:
      title: Media Image
      description: The image for the component.
      type: string
    imagepositon:
      type: string
      title: Image Position
      description: The image position.
    topics:
      type: string
      title: Topics
    date:
      type: string
      title: Date
    links:
      type: string
      title: Links
```

05

Benefits of Using Complementary Modules with SDC

- Integrating additional modules with SDC opens doors to a world of possibilities.
- Save time, improve collaboration, and ensure compliance with accessibility standards, all while delivering richer frontend experience.

Benefits of Using Complementary Modules with SDC

What are the benefits of using sdc_display?

Performance Optimization

"It's like giving your site a turbo boost!"

Easy to maintain and build

"Making frontend work so easy, even your grandma could do it!"

Save time and costs

Because who doesn't love a superhero on a budget?

In the Drupal universe, our toolkit isn't just a spaceship;
we're the quirky engineers, zapping time-wasting aliens, turbo-charging website warp drives,
and turning complicated code into a walk in the park! All while saving time, costs, and our sanity!

06

Future Perspectives: Predicting the Evolution and Impact of SDC in the Drupal Community

- Updated journey of a Drupal Frontend Developer
- What SDC **doesn’t** solve
- Developers with different backgrounds
- Developers with experience in Drupal
- Companies and clients
- Drupal community

Updated journey of a Drupal Frontend Developer

Start

After modelling the requirements, and creating the necessary entities, view modes, etc.

Starting work on a **component**.

Finding the necessary template files.

~~Attaching the necessary libraries.~~

Coding the necessary markup, styling and JS.

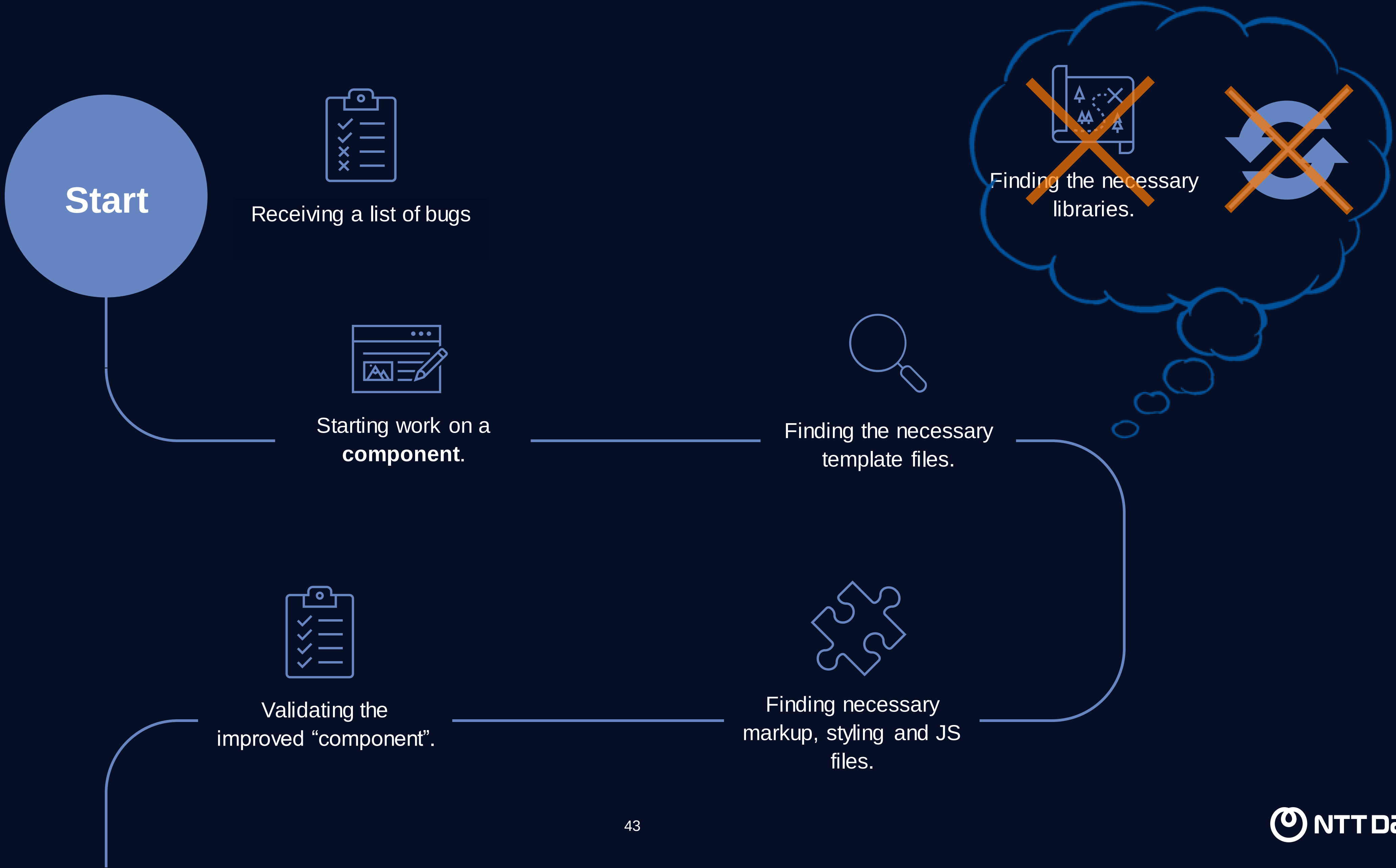
Validating the new **component**.

Repeat for all site components

(variations of previously developed ones might be more easily approached with the technique)

NTT DaTa

Updated journey of a Drupal Frontend Developer: Debugging


Start
Receiving a list of bugs
Finding the necessary libraries.
Starting work on a **component**.
Finding the necessary template files.
Validating the improved “component”.
Finding necessary markup, styling and JS files.

What SDC doesn't solve

Start

After modelling the requirements, and creating the necessary entities, view modes, etc.

Starting work on a **component**.

Finding the necessary template files.

Someone still needs to know where to place the components

Coding the necessary markup, styling and JS.

Validating the new **component**.

Repeat for all site components

(variations of previously developed ones might be more easily approached with the technique)

Impact: Developers with different backgrounds

- Can focus on developing modular components
- Don’t have to worry about the Drupalisms
- Can focus on providing new functionality
- Don’t have to worry about where the components will be rendered
- Files for the component are all together

Impact: Developers with experience in Drupal

- Can focus on using (and reusing) the developed components across multiple templates – render the components where necessary
- Can collaborate on components across different projects
- Can focus on integrating different components
- Easier debugging experience (files for the component are all together)

Impact: Companies and clients

Companies: next steps

- Building up a library of common components which can be reused across projects

Companies: benefits

- Working from a library of components => project accelerators
- Easier to maintain code quality across projects

Clients: benefits

- Working from a library of components => faster time to Go Live
- Potential for less repeated code and less general code => faster websites

Impact: Drupal Community

SDC in Drupal core >= 10.1

- Is now Included in core
- Possibility to create a community-based library of components
- Components can be overridden in themes

=> Contrib themes could start implementing their own components which could be overridden in custom themes that extend them

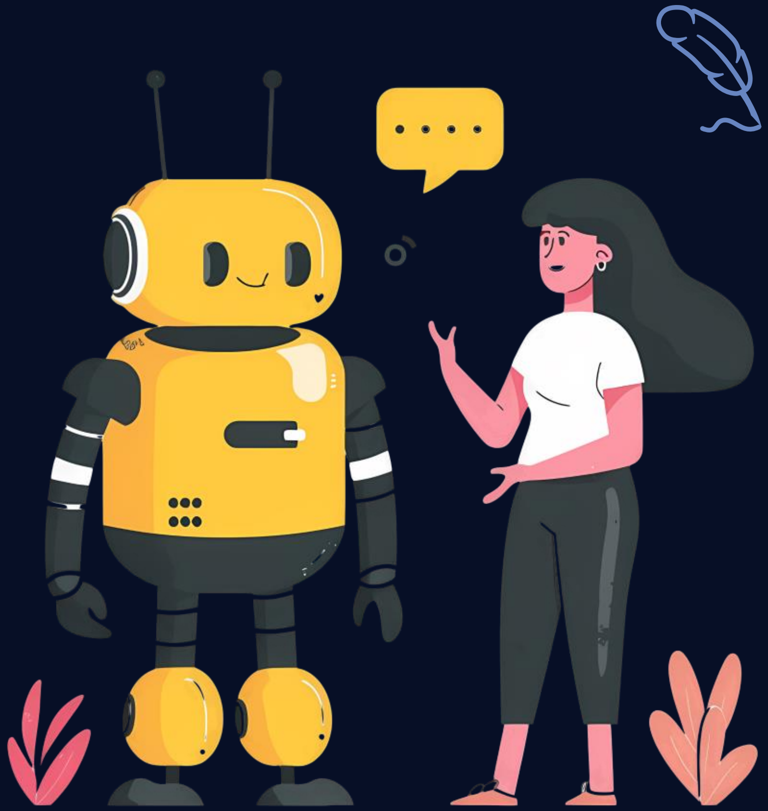
SDC Display is a contrib module

Works with Drupal >=10.1

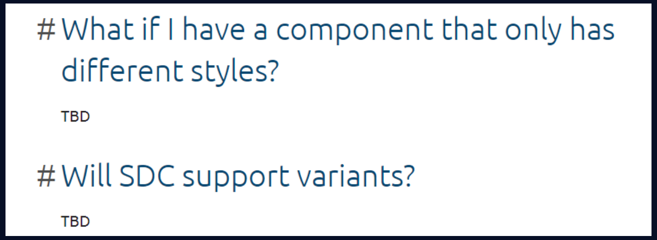
The community can start using the module and providing feedback => improvements

- Module is now on 1.0.0-beta5 released *20 March 2024*

SDC: Next steps



- Possible News integration with SDC
- Community contributions
- Focus on documentation
- Define best practices
- Questions still to answer (The last update was in April this year)

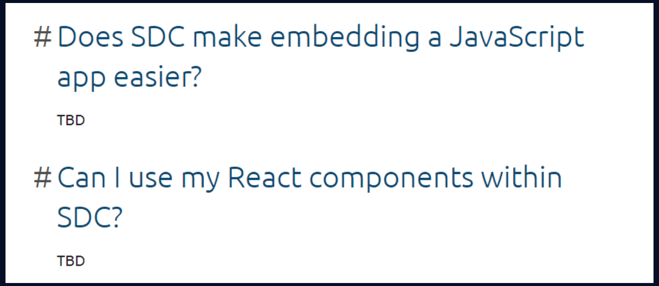


What if I have a component that only has different styles?

TBD

Will SDC support variants?

TBD



Does SDC make embedding a JavaScript app easier?

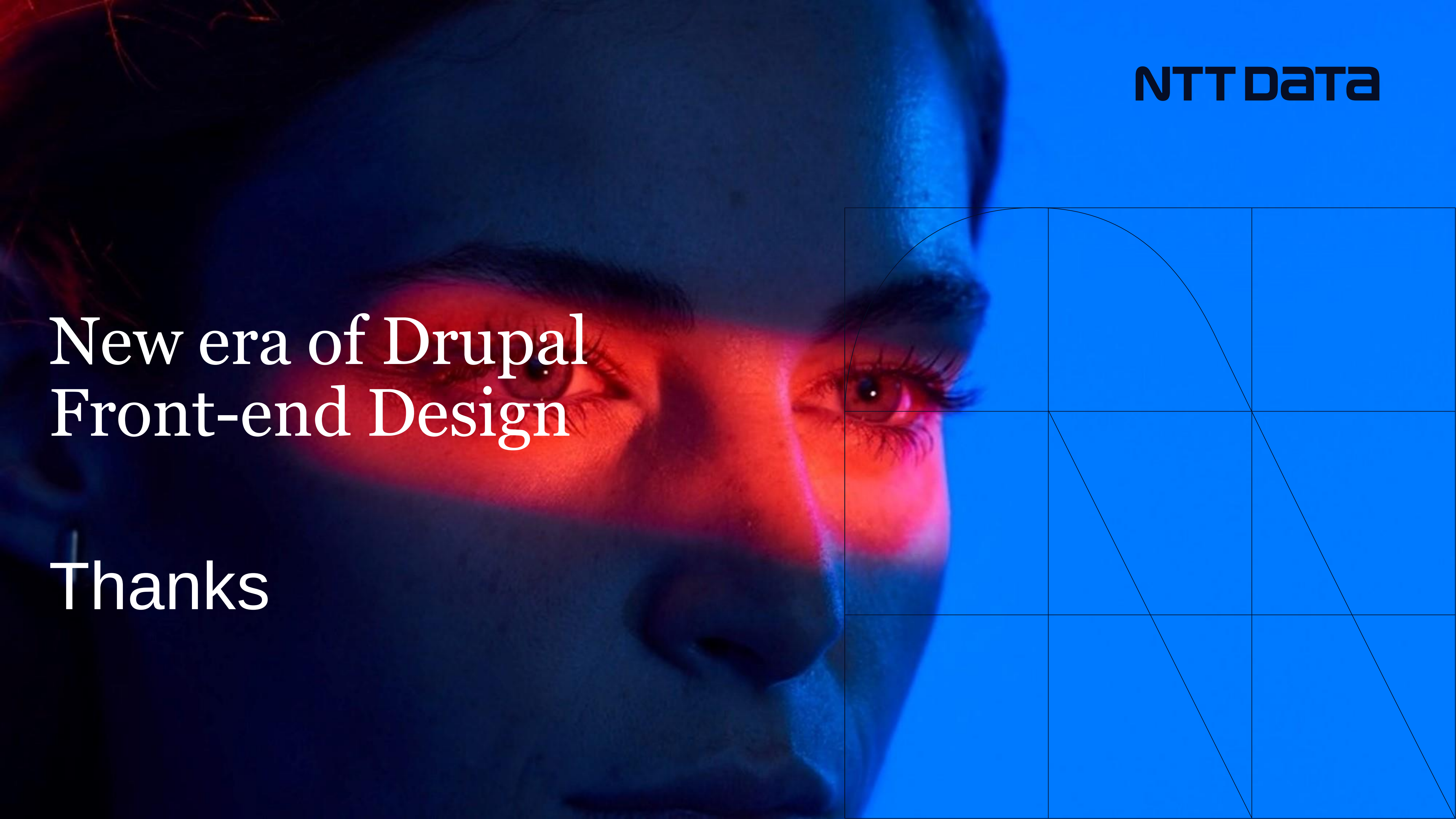
TBD

Can I use my React components within SDC?

TBD

SDC: Learn more

- Project on Drupal.org - https://www.drupal.org/project/sdc
- Drupal.org SDC development info - https://www.drupal.org/docs/develop/theming-drupal/using-single-directory-components/about-single-directory-components
- Lullabot article - https://www.lullabot.com/articles/getting-single-directory-components-drupal-core
- Herchel article - https://herchel.com/articles/creating-your-first-single-directory-component-within-drupal
- Specbee article - https://www.specbee.com/blogs/component-based-theming-with-drupal-single-directory-component
- Axelerant article - https://www.axelerant.com/blog/single-directory-components-in-drupal
- Drupal: Converting a component to Single Directory Components (SDC) [Youtube] - https://youtu.be/DbpZOhiq_Ho?feature=shared
- Single Direct Components Display - https://www.drupal.org/project/sdc_display
- Components as Site Building Tools - https://video.mateuaguilo.com/w/sC5nv52GLQrPHYcjBUvEeN



NTT DATA

New era of Drupal Front-end Design

Thanks

NTT DATA