

# Les deux Render API

Apprécier la “bonne” et éviter la “mauvaise”





Degemer mat 🖐️

[/u/pdureau](#)

Bayonne - Paris - Bruxelles

Drupal since 2006

Design systems & WebAssembly lover









# ATTENTION CES SLIDES NE SONT PAS DES SLIDES SUR LA FORM API

MERCI DE VOTRE  
COMPREHENSION

# Part 1: Render API?





# Once upon a time in summer 2006

## Building `$node->body` with arrays like FAPI for viewing



**eaton** Credit commented *26 July 2006 at 07:04*

#1

RobRoy, I've been giving some thought to this and if it's done right, I think it can be a huge improvement. I'll brain-dump my thinking on the subject so far and perhaps we'll come up with something that works.

1. The solution should allow module-provided data to be kept separate from rendered HTML
2. The solution should preserve the original body and teaser values from the database (if any) for use by themers
3. The solution should allow individual pieces of node data *or* the entire assembled node to be filtered and/or processed for viewing
4. The fully rendered teaser and body should still be stored in `$node->teaser` and `$node->body` once everything has been processed.
5. The solution should mirror existing Drupal systems (notably FormAPI) where possible. A new 'style' of data structures is bad.

**Closed (fixed)**

**Project:** Drupal core

**Version:** 5.x-dev

**Component:** node system

**Priority:** Normal

**Category:** Task

**Assigned:** [eaton](#)

**Reporter:** [RobRoy](#)

**Created:** 18 Jul 2006 at 20:41 CEST

**Updated:** 8 Dec 2006 at 22:46 CET



# So easy to use

Just return an array from the expected method or function

```
return [  
  '#type' => link,  
  '#title' => “Degemer mat 🖐️”,  
  '#url' => new Url('https://e.org'),  
];
```

[RenderableInterface::toRenderable\(\)](#)

[LayoutInterface::build\(\)](#)

[FormatterInterface::view\(\)](#)

[FormatterInterface::viewElements\(\);](#)

[BlockPluginInterface::build\(\)](#)

....





# So powerful

## Declarative

Easy to type. (de)Serializable if clean.

## Easy nesting

The Virtual DOM of Drupal. We are building a tree.

## Data bubbling

Declare locally, impact globally

## Asset libraries management

Our beloved libraries.yml

## Clever caching

Context, tags, keys...

```
[
  "#theme" => "item_list",
  "#items" => [
    [ "value" =>
      [
        "#theme" => "image",
        "#uri" => "/path/to/image"
      ]
    ]
  ],
  '#attached' => [
    'library' => ['vendor/lib']
  ],
  '#cache' => [
    'contexts' => ['user']
  ]
]
```



# We love it.. do we?



*"I still feel bad as one of the earliest proponents of abstracting FormAPI into a generic RenderAPI."*

Jeff Eaton, 2012

in [john.albin.net/./arrays-of-doom](http://john.albin.net/./arrays-of-doom)

Always the same feedbacks since 2012:

- Not discoverable
- Too much Drupalism
- **Too weirdly complex**

Is it inherent or accidental complexity?

A wooden table covered with various art supplies. In the foreground, there are two blank white sheets of paper. To the left, a paint palette with brushes and a palette knife is visible. In the center, there is a watercolor palette with various colors. To the right, there are several containers of paint, brushes, and a fan of green paper. The table is cluttered with paint splatters and other art materials.

**Part 2: What a mess!**



# 3 types of renderables

## 34 render elements

```
$ grep -hoEr "#type" => '(\S+)' core/ --exclude-dir  
tests | sort | uniq -c | sort -nr
```

```
138 #type' => 'details'  
94 #type' => 'container'  
93 #type' => 'link'  
68 #type' => 'table'  
47 #type' => 'actions'  
36 #type' => 'inline_template'  
27 #type' => 'fieldset'  
23 #type' => 'html_tag'  
21 #type' => 'status_messages'  
14 #type' => 'pager'  
...
```

## 99 theme hooks

```
$ grep -hoEr "#theme" => '(\S+)' core/ --exclude-dir  
tests | grep -v "._" | sort | uniq -c | sort -nr
```

```
84 #theme' => 'item_list'  
21 #theme' => 'username'  
16 #theme' => 'image'  
7 #theme' => 'status_messages'  
7 #theme' => 'image_style'  
6 #theme' => 'table'  
6 #theme' => 'links'  
5 #theme' => 'update_version'  
5 #theme' => 'indentation'  
5 #theme' => 'file_upload_help'  
...
```

## 2 special ones

```
1016 '#markup'  
85 '#plain_text'
```



# 135 renderables to learn?



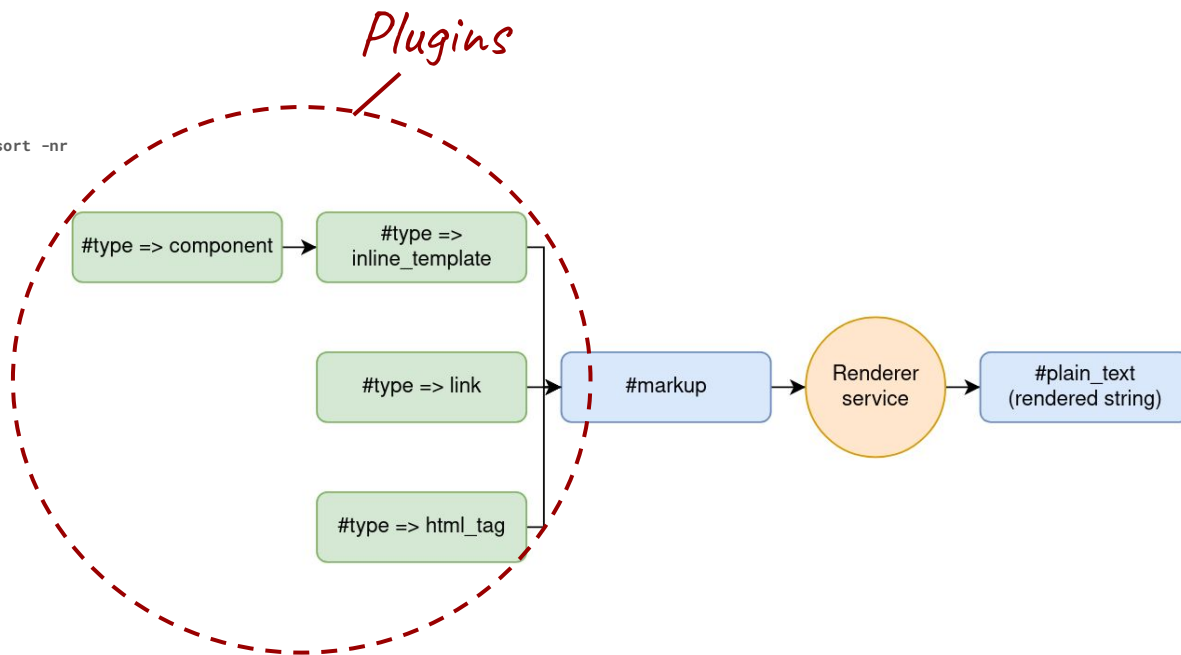


# 4 render elements are directly rendered

## 34 render elements

```
$ grep -hoEr "#type' => '(\S+)'" core/ | sort | uniq -c | sort -nr
```

```
138 #type' => 'details'  
94 #type' => 'container'  
93 #type' => 'link'  
68 #type' => 'table'  
47 #type' => 'actions'  
36 #type' => 'inline_template'  
27 #type' => 'fieldset'  
23 #type' => 'html_tag'  
21 #type' => 'status_messages'  
...  
6 #type' => 'component'
```





# The others are wrappers around theme hooks

## 34 render elements

```
$ grep -hoEr "#type' => '(\S+)'" core/ | sort | uniq -c | sort -nr
```

138 #type' => 'details' (+ library attach.)

94 #type' => 'container'

93 #type' => 'link'

68 #type' => 'table' (+ library attach.)

47 #type' => 'actions'

36 #type' => 'inline\_template'

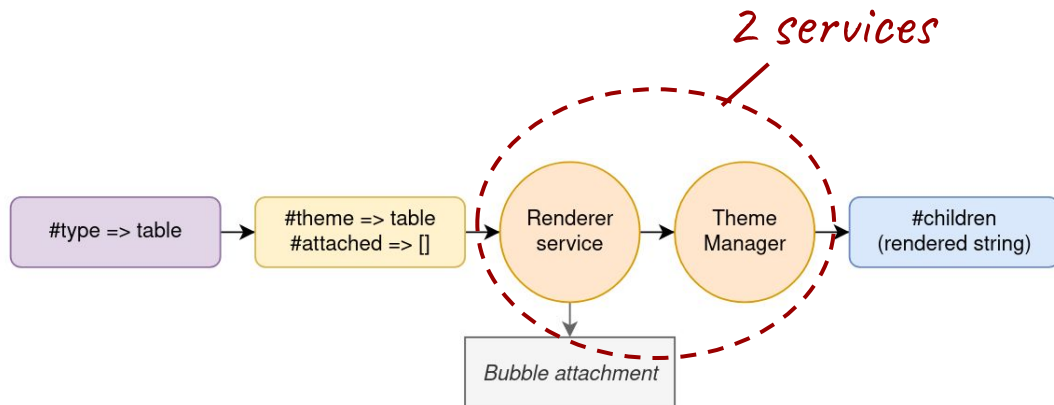
27 #type' => 'fieldset'

23 #type' => 'html\_tag'

21 #type' => 'status\_messages'

14 #type' => 'pager'

...





~~135~~ 36 renderables to learn?







# Why do we need to wrap #theme?

#theme are so old...

Issues:

- Straightforward call to theme()
- Definitions hardly extensible and alterable.
- Can't *prepare* data, including attachment

#type should actually be the "de facto" starting point...

But some #theme have no corresponding #type:

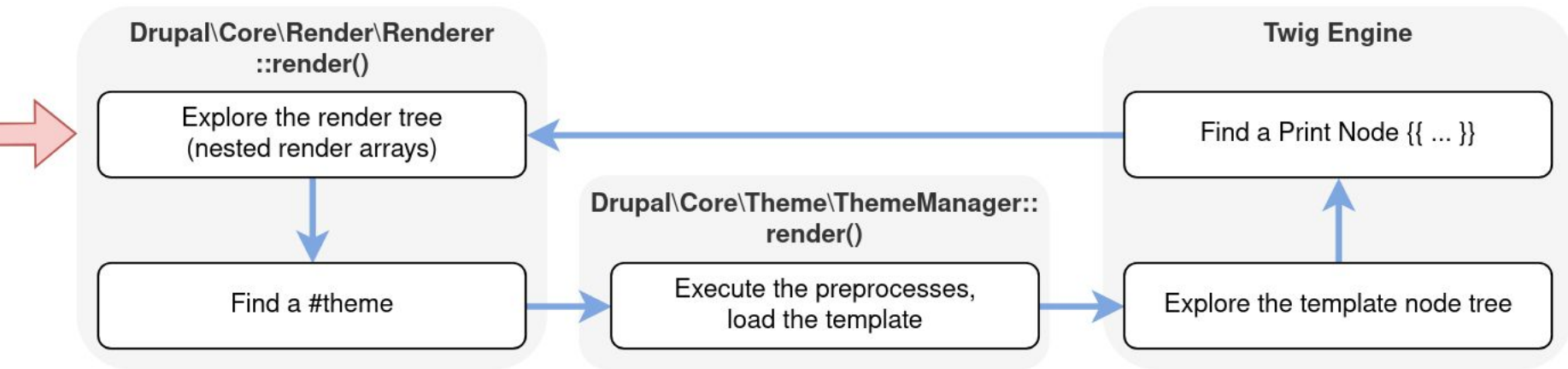
- Breadcrumb
- Progress bar
- Links
- ...

A painting depicting three artists in a lush, green field. In the foreground, a man in a dark blue jacket and cap sits on a wooden stool, painting on an easel. To his right, another man in a light-colored jacket and a wide-brimmed hat sits on a similar stool, also painting. In the background, a woman in a blue dress and white headscarf sits on a stool, painting. The scene is set in a vibrant, sunlit landscape with tall grass and trees. The overall style is characteristic of the Impressionist movement.

# Part 3: A tale of 2 renderers



# if #theme is found...





# The good stuff is in Render\Renderer

## Declarative

Easy to type. (de)Serializable if clean.

## Easy nesting

The Virtual DOM of Drupal. We are building a tree.

## Data bubbling

Declare locally, impact globally

Asset **libraries** management

Our beloved libraries.yml

## Clever caching

Context, tags, keys...

The good:

- #markup & #plain\_text
- #attached
- #cache
- lazy\_builder & placeholders

The bad: #prefix & #suffix

The ugly: #pre\_render



# So, what is ThemeManager::render() for?

## The good:

- default 'attributes'
- Template loading

## The bad

### *Theme wrappers*

Confusing and useless. You can always use an upper level instead.

## The ugly

### *Hooks suggestions*

Not discoverable. Messy. Blur the business / view separation.

### *Preprocess hooks*

Risky. Unfriendly. Blur the business / view separation. Unpredictable order of execution.

# Part 4: Can we skip the ThemeManager?



# Do you know Single Directory Components ?

UI Component API. Si

And more:

Can do everything

Easy YML declaration

Like hook themes

ing (slots & props)

- Load templates
- Inject attributes

gh the

Like render elements:

- Library asset attachment

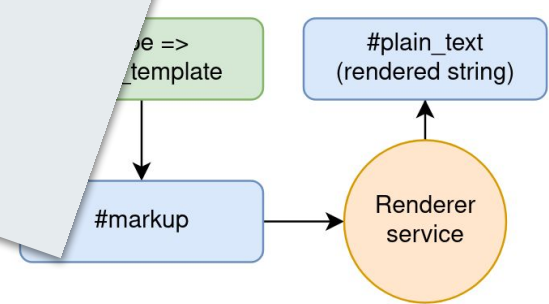
**Single Directory Component : de l'atomic design natif dans Drupal**

Harelle Quentin

40 min

Intermédiaire

Intégration, méthodologie



# Standard profile's front page

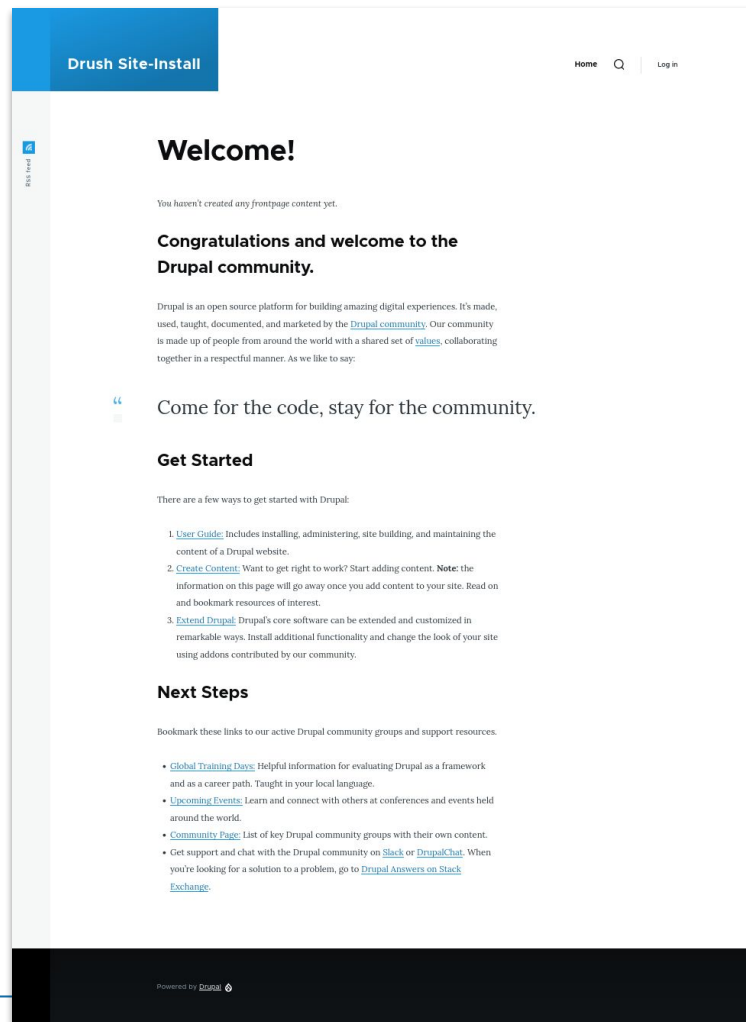
*In anonymous browsing.*

26 renderables

- 2 render elements
- 19 theme hooks or wrappers
- 5 #markup or #plain\_text

22 calls to ThemeManager::render()

Perf: 650ms after cache flush



The screenshot shows the 'Welcome!' page of a Drupal site during installation. The page has a blue header with 'Drush Site-Install' and navigation links for 'Home', 'Q', and 'Log in'. A small Drupal logo is in the top right corner. The main content area features a 'Welcome!' heading, a note that no frontpage content has been created yet, and a congratulatory message to the Drupal community. It includes a quote: 'Come for the code, stay for the community.' Below this is a 'Get Started' section with three numbered links: 'User Guide', 'Create Content', and 'Extend Drupal'. A 'Next Steps' section follows, with a heading 'Bookmark these links to our active Drupal community groups and support resources.' and a list of links for 'Global Training Days', 'Upcoming Events', and 'Community Page'. The footer of the page is black with the text 'Powered by Drupal' and the Drupal logo.

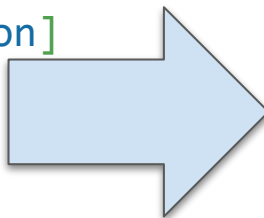






# 1. Replace #theme\_wrappers by a proper wrapper

```
"#type": page
header:
  elements:
    "#theme_wrappers": [region]
    "#region": header
    block_1: {}
    block_2: {}
```



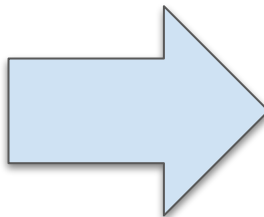
```
"#type": page
header:
  "#theme": region
elements:
  block_1: {}
  block_2: {}
```

Not working but  
needed



## 2. Remove blocks layer

```
"#theme": block
"#plugin_id": page_title_block
"#base_plugin_id": page_title_block
"#id": olivero_page_title
content:
  "#type": page_title
  "#title":
    "#markup": Welcome!
```



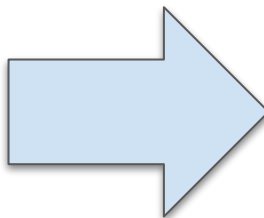
```
"#type": page_title
"#title":
  "#markup": Welcome!
```

Some CSS target  
blocks wrappers



### 3. Replace remaining #theme by SDC

```
"#theme": feed_icon  
"#url": ""  
"#title": RSS feed
```



```
"#type": component  
"#component": "dc:feed_icon"  
"#slots":  
  title: RSS feed  
"#props":  
  url: ""
```

Component definition

```
name: Feed icon  
slots:  
  title:  
    title: Title  
props:  
  type: object  
  properties:  
    url:  
      title: Url  
      type: string  
      format:  
        iri-reference
```



# Results

## Before

26 renderables

- 2 render elements
- 19 theme hooks or wrappers
- 5 #markup or #plain\_text

22 calls to ThemeManager::render()

Perf: 650ms after cache flush

## ✓ After

13 renderables

- 10 render elements
- 0 theme hooks
- 3 markup or plain\_text







0 calls to ThemeManager::render()

Perf: 600ms after cache flush



# See by yourself

<https://github.com/pdureau/drupal-camp-fr-2024>

 <b>pdureau</b>	Add revised renderable	d63edd4 · now	 2 Commits
 components	Add revised rendera...		now
 src/Controller	Add revised rendera...		now
 drupal_camp.info.yml	Add revised rendera...		now
 drupal_camp.routin...	Add original rendera...		1 hour ago

A photograph of a field of white flowers, possibly baby's breath, at sunset. The sun is low on the horizon, creating a warm, golden glow. The foreground shows several tall, thin stems with clusters of small white flowers. The background is a vast field of similar flowers stretching to the horizon. The text "Part 5: Conclusion" is overlaid in the center in a white, sans-serif font.

## Part 5: Conclusion



# 6 renderables to learn!





# Yes, 6 renderables

## **#type=component**

#component

#slots

#props

## **#type=inline\_template**

#context

## **#type=html\_tag**

#tag

#value

## **#type=link**

#url

#title

## **#markup**

#allowed\_tags

## **#plain\_text**

With shared properties: #attached,  
#cache, #attributes...





# The direction of history

*Direct call to templates.  
Can it be all so simple?*

**Drupal 5**  
Simple nesting  
of theme()  
functions  
#theme,  
#prefix, #suffix,  
#weight...

**Drupal 6**  
theme()  
functions need  
extra  
processing  
#pre\_render &  
#post\_render

*Late rendering is a  
must-have*

**Drupal 7**  
theme()  
functions are  
rendered too  
early  
#theme\_wrap  
pers & #type

*UI logic in templates  
instead of PHP*

**Drupal 8**  
“Consensus  
Banana”:  
processing  
moved from  
PHP to Twig

**Drupal 10**  
SDC in core.  
Well defined  
models (slots &  
props)

*UI components*



# Enjoy an easier render API

Theme hooks will not disappear from core. At least not soon.

Some of them will not be suitable for conversion to SDC anyway:  
block, node, view, field...

ThemeManager mechanisms (template suggestions, preprocess hooks...) are not avoidable because of core and contrib use.

But for your custom development, you can skip them and enjoy a simpler and easier Render API.



**Merci**  
*pour votre*  
*écoute !*

