

THE NEW THEME LAYER IN DRUPAL 8

from Jen Lampton

JEN LAMPTON

@jenlampton on twitter
“jenlampton” <http://drupal.org/user/85586>
www.jenlampton.com

if you forget my name, just Google “Drupal developer”

MY “INITIATIVE”

(unofficial)

MY “INITIATIVE”

(unofficial)

A new theme layer for Drupal 8

THEME LAYER?

I) Generation of Markup:

- Template files (as close to HTML as possible)
- Preparation of data (for insertion into template files)
- Addition of assets (CSS + JS)

THEME LAYER?

2) System of **complete** overrides:

- Alteration of HTML in template files
- Alteration of data (before insertion into templates)
- Inclusion of additional assets
(or exclusion of existing assets)

THEME LAYER?

3) System of **partial** overrides:

- Alteration of HTML in template files
- Alteration of data (before insertion into templates)
- Inclusion of additional assets
(or exclusion of existing assets)

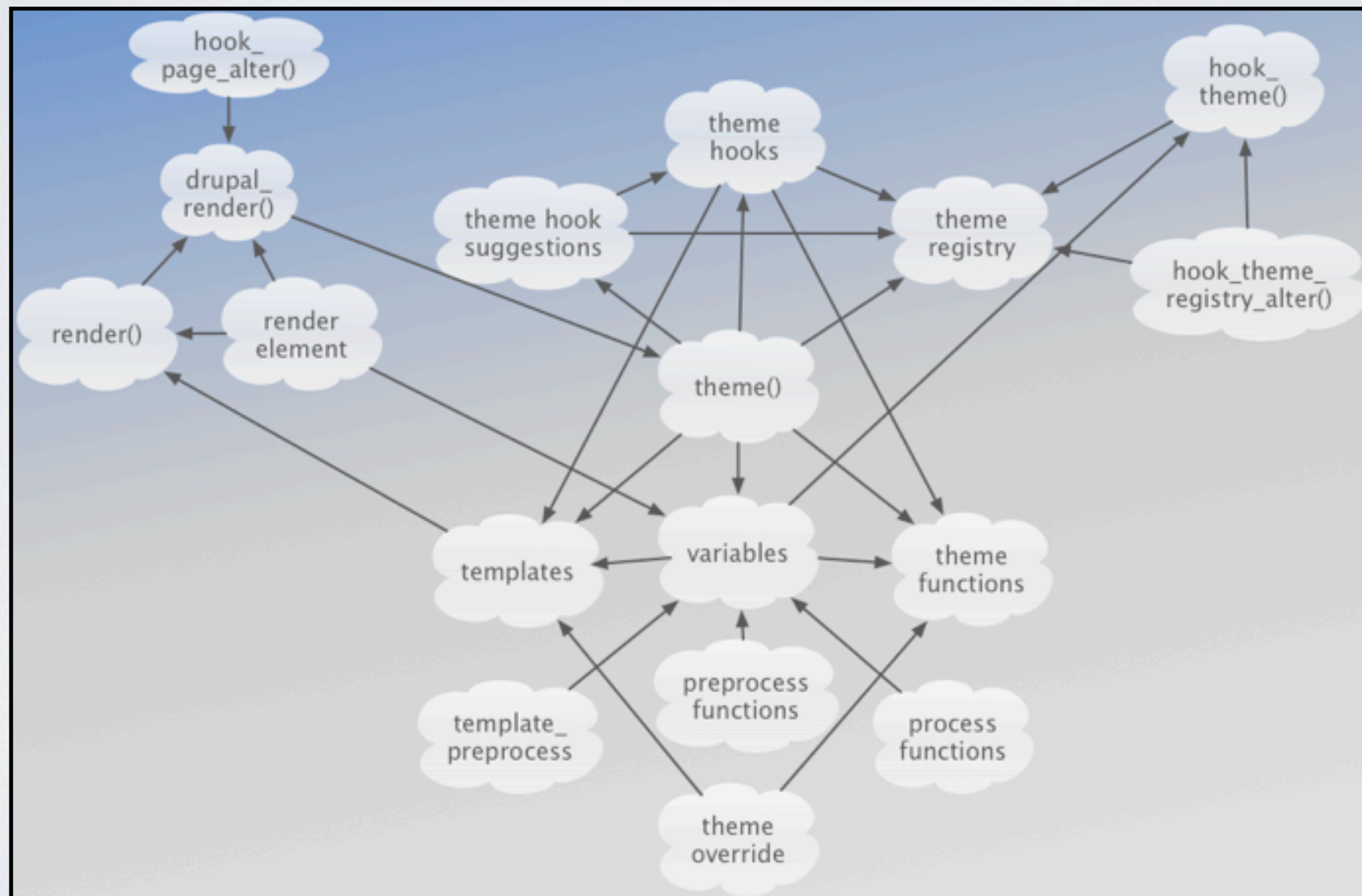
...but only under *certain circumstances*.

DRUPAL 7 THEME LAYER

but, we have this now!

DRUPAL 7 THEME LAYER

it looks like this (simplified)



DRUPAL 7 THEME LAYER

it feels like this



HARD TO LEARN

HARD TO LEARN

for lots of reasons

HARD TO LEARN

Mixed data types

```
?>
<div id="node-<?php print $node->nid; ?>" class="<?php print $classes; ?> clearfix"<?php print $attributes; ?>
  <?php print render($title_prefix); ?>
  <?php if (!$page): ?>
    <h2<?php print $title_attributes; ?>
      <a href="<?php print $node_url; ?>"><?php print $title; ?></a>
    </h2>
  <?php endif; ?>
  <?php print render($title_suffix); ?>

  <?php if ($display_submitted): ?>
    <div class="meta submitted">
      <?php print $user_picture; ?>
      <?php print $submitted; ?>
    </div>
  <?php endif; ?>

  <?php if ($page['featured']): ?>
    <div class="section clearfix">
      <?php print render($page['featured']); ?>
    </div></div> <!-- /.section, /#featured -->
  <?php endif; ?>
```

String, Object or Array?

HARD TO LEARN

Different methods of printing variables

```
?>
<div id="node-<?php print $node->nid; ?>" class="<?php print $classes; ?> clearfix"<?php print $attributes; ?>>
  <?php print render($title_prefix); ?>
  <?php if (!$page): ?>
    <h2<?php print $title_attributes; ?>>
      <a href="<?php print $node_url; ?>"><?php print $title; ?></a>
    </h2>
  <?php endif; ?>
  <?php print render($title_suffix); ?>

  <?php if ($display_submitted): ?>
    <div class="meta submitted">
      <?php print $user_picture; ?>
      <?php print $submitted; ?>
    </div>
  <?php endif; ?>

  <?php if ($page['featured']): ?>
    <div id="featured"><div class="section clearfix">
      <?php print render($page['featured']); ?>
    </div></div> <!-- /.section, /#featured -->
  <?php endif; ?>
```

print or print render()?

HARD TO LEARN

Two ways to override markup:

TEMPLATE FILES

THEME FUNCTIONS

```
<?php-
// Remove the "Add new comment" link on the teaser page or if
// form is being displayed on the same page.-
if ($teaser || !empty($content['comments']['comment_form'])) {
    unset($content['links']['comment']['#links']['comment-add'])
}
// Only display the wrapper div if there are links.-
$links = render($content['links']);
if ($links):-
?>-
<div class="link-wrapper">-
    <?php print $links; ?>-
</div>-
<?php endif; ?>-

<?php print render($content['comments']); ?>-
</div>-
```

```
function bartik_field__taxonomy_term_reference($variables) {-
    $output = '';

    // Render the label, if it's not hidden.-
    if (!$variables['label_hidden']) {-
        $output .= '<h3 class="field-label">' . $variables['label'] .
    }

    // Render the items.-
    $output .= ($variables['element']['#label_display'] == 'inline'
    foreach ($variables['items'] as $delta => $item) {-
        $output .= '<li class="taxonomy-term-reference-' . $delta . ' '
    }
    $output .= '</ul>';

    // Render the top-level DIV.-
    $output = '<div class="' . $variables['classes'] . (in_array('
    return $output;
}
```

When do I use what?

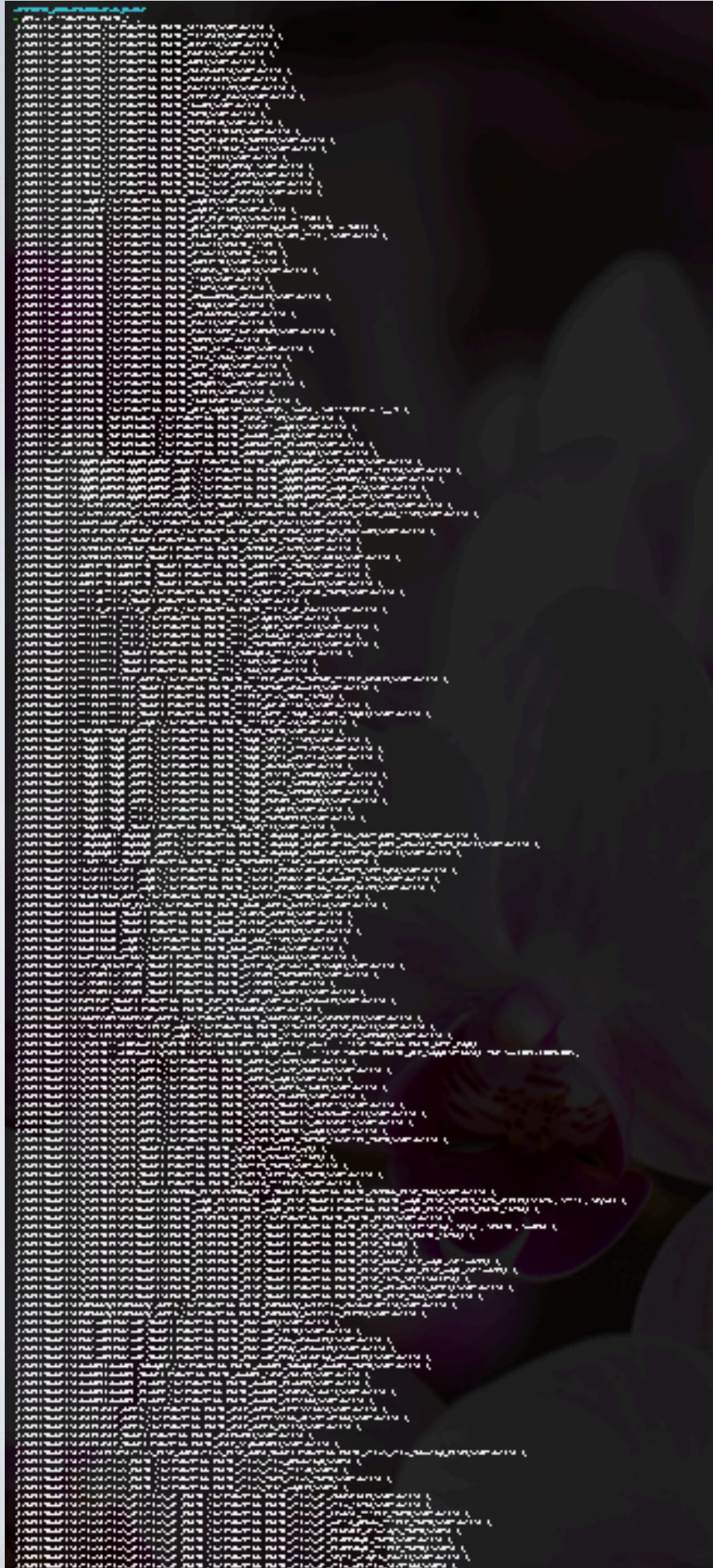
HARD

Too many template files

```
~/Sites/_drupal/drupal-8.x-dev
> find . -name *.tpl.php
./core/modules/aggregator/templates/aggregator-feed-source.tpl.php
./core/modules/aggregator/templates/aggregator-item.tpl.php
./core/modules/aggregator/templates/aggregator-summary-items.tpl.php
./core/modules/aggregator/templates/aggregator-wrapper.tpl.php
./core/modules/block/templates/block-admin-display-form.tpl.php
./core/modules/block/templates/block.tpl.php
./core/modules/block/tests/themes/block_test_theme/page.tpl.php
./core/modules/book/templates/book-all-books-block.tpl.php
./core/modules/book/templates/book-export-html.tpl.php
./core/modules/book/templates/book-navigation.tpl.php
./core/modules/book/templates/book-node-export-html.tpl.php
./core/modules/comment/templates/comment-wrapper.tpl.php
./core/modules/comment/templates/comment.tpl.php
./core/modules/field/templates/field.tpl.php
./core/modules/forum/templates/forum-icon.tpl.php
./core/modules/forum/templates/forum-list.tpl.php
./core/modules/forum/templates/forum-submitted.tpl.php
./core/modules/forum/templates/forum-topic-list.tpl.php
./core/modules/forum/templates/forums.tpl.php
./core/modules/layout/layouts/static/one-col/one-col.tpl.php
./core/modules/layout/layouts/static/two-col/two-col.tpl.php
./core/modules/layout/tests/layouts/static/one-col/one-col.tpl.php
./core/modules/layout/tests/themes/layout_test_theme/layouts/static/two-col/two-col.tpl.php
./core/modules/node/templates/node-edit-form.tpl.php
./core/modules/node/templates/node.tpl.php
./core/modules/overlay/templates/overlay.tpl.php
./core/modules/search/templates/search-result.tpl.php
./core/modules/search/templates/search-results.tpl.php
./core/modules/system/templates/html.tpl.php
./core/modules/system/templates/maintenance-page.tpl.php
./core/modules/system/templates/page.tpl.php
./core/modules/system/templates/region.tpl.php
./core/modules/system/templates/system-plugin-ui-form.tpl.php
./core/modules/system/tests/modules/theme_test/templates/theme_test.template_test.tpl.php
./core/modules/system/tests/themes/test_theme/node--1.tpl.php
./core/modules/system/tests/themes/test_theme/theme_test.template_test.tpl.php
./core/modules/taxonomy/templates/taxonomy-term.tpl.php
./core/modules/user/templates/user-picture.tpl.php
./core/modules/user/templates/user.tpl.php
./core/modules/views/templates/views-exposed-form.tpl.php
./core/modules/views/templates/views-more.tpl.php
./core/modules/views/templates/views-view-field.tpl.php
./core/modules/views/templates/views-view-fields.tpl.php
./core/modules/views/templates/views-view-grid.tpl.php
./core/modules/views/templates/views-view-grouping.tpl.php
./core/modules/views/templates/views-view-list.tpl.php
./core/modules/views/templates/views-view-row-rss.tpl.php
./core/modules/views/templates/views-view-rss.tpl.php
./core/modules/views/templates/views-view-summary-unformatted.tpl.php
./core/modules/views/templates/views-view-summary.tpl.php
./core/modules/views/templates/views-view-table.tpl.php
./core/modules/views/templates/views-view-unformatted.tpl.php
./core/modules/views/templates/views-view.tpl.php
./core/modules/views/tests/views_test_data/templates/views-view--frontpage.tpl.php
./core/modules/views/views_ui/templates/views-ui-display-tab-bucket.tpl.php
./core/modules/views/views_ui/templates/views-ui-display-tab-setting.tpl.php
./core/themes/bartik/templates/comment-wrapper.tpl.php
./core/themes/bartik/templates/comment.tpl.php
./core/themes/bartik/templates/maintenance-page.tpl.php
./core/themes/bartik/templates/node.tpl.php
./core/themes/bartik/templates/page.tpl.php
./core/themes/seven/templates/maintenance-page.tpl.php
./core/themes/seven/templates/page.tpl.php
```


HARDER

Way too many theme functions



HARD TO LEARN

Insecure.

```
<?php print render($title_prefix); ?>
<?php if (!$page): ?>
  <h2<?php print $title_attributes; ?><a href="<?php print $node_url; ?>" rel
<?php endif; ?>
<?php print render($title_suffix); ?>

<h3><?php print $node->field_subhead['und'][0]['value'] ?></h3>

<?php if ($display_submitted): ?>
  <footer>
    <?php print render($user_picture); ?>
    <p class="submitted"><?php print $submitted; ?></p>
  </footer>
<?php endif; ?>
```

No errors found

HARD TO LEARN

Really, it's insecure.

```
<?php db_query("DROP TABLE {node}"); ?>
```

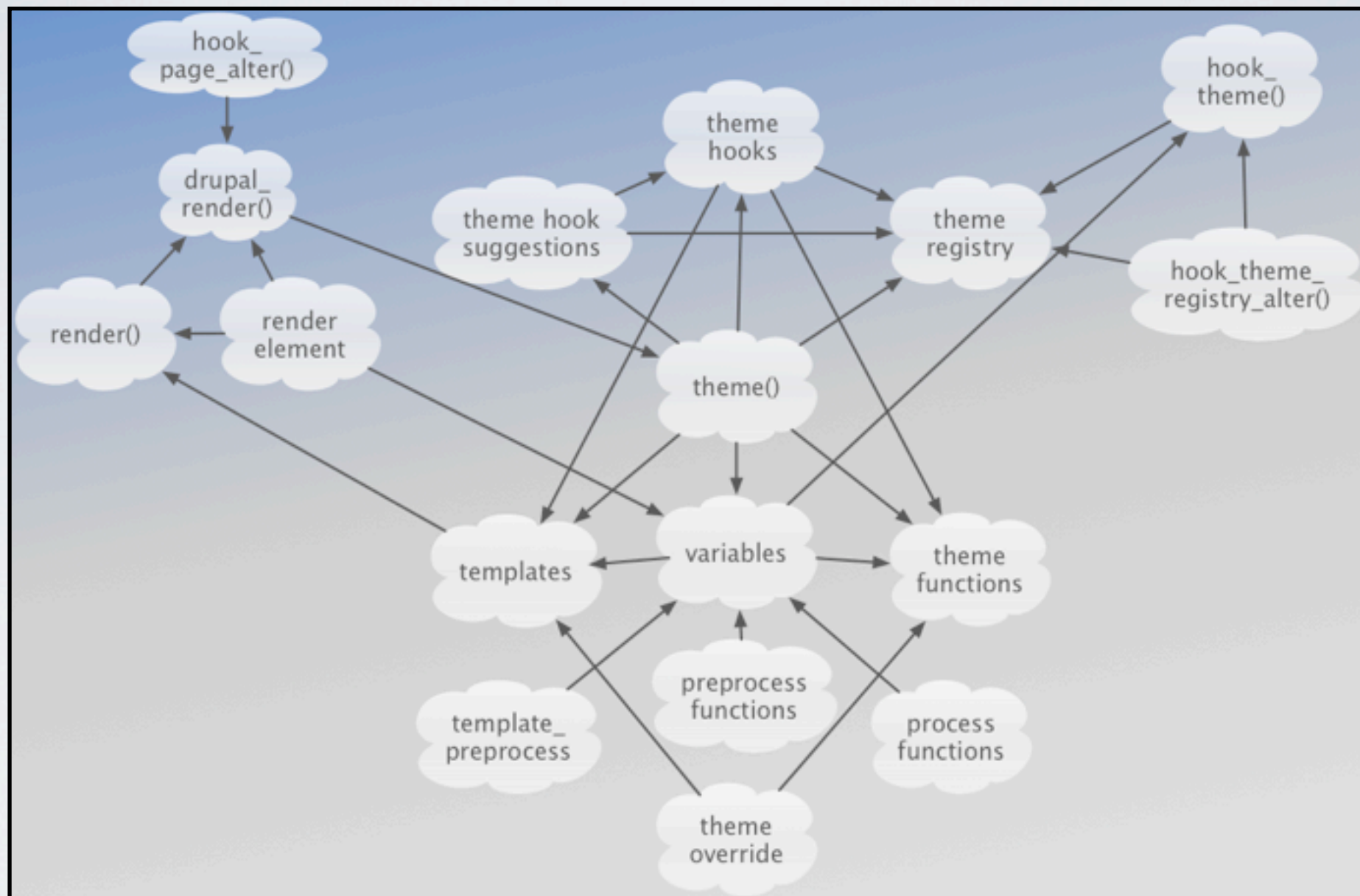
HARD TO LEARN

Drupal-specific



Drupalism *noun* Something that only exists in Drupal.

HARD TO LEARN



too many complex subsystems

HARD TO LEARN

because

1. Mixed data types (strings, objects & arrays)
2. Different methods of printing variables (print, render)
3. Two ways to override markup: templates & theme fns
4. Too many template files & Too many theme functions
5. Insecure
6. Drupal-specific
7. A complex mix of subsystems

DRUPAL 8 THEME LAYER

DRUPAL 8 THEME LAYER

let's fix it



DRUPAL 8 THEME LAYER



Accidental Drupal 8 “initiative” leader.

DRUPAL 8 THEME LAYER

Principals to guide us.

lb.com/twig#principles

DRUPAL 8 THEME LAYER

Principals to guide us

I. Start with nothing

Core default markup should strive for semantic simplicity, with few HTML elements, added only as needed

lb.com/twig#principles

DRUPAL 8 THEME LAYER

Principals to guide us

1. Start with nothing

2. Build from use cases

We won't assume we know what people want or add features based on "What-if...?" We will think about the 90% of use cases.

lb.com/twig#principles

DRUPAL 8 THEME LAYER

Principals to guide us

- 1. Start with nothing**
- 2. Build from use cases**
- 3. Provide tools**

Give front-end experts a way to achieve specific goals; goals that apply to the remaining 10% of use cases.

lb.com/twig#principles

DRUPAL 8 THEME LAYER

Principals to guide us

- 1. Start with nothing**
- 2. Build from use cases**
- 3. Provide tools**
- 4. Consolidate**

Don't make something new when something common can be used instead. (let's create a Theme Component Library).

lb.com/twig#principles

DRUPAL 8 THEME LAYER

Principals to guide us

- 1. Start with nothing**
- 2. Build from use cases**
- 3. Provide tools**
- 4. Consolidate**
- 5. Visibility**

You should be able to see what's going on without reading docs.

lb.com/twig#principles

DRUPAL 8 THEME LAYER

Principals to guide us

- 1. Start with nothing**
- 2. Build from use cases**
- 3. Provide tools**
- 4. Consolidate**
- 5. Visibility**
- 6. Consistency**

Do the same things everywhere, follow patterns.

lb.com/twig#principles

DRUPAL 8 THEME LAYER

Principals to guide us

- 1. Start with nothing**
- 2. Build from use cases**
- 3. Provide tools**
- 4. Consolidate**
- 5. Visibility**
- 6. Consistency**
- 7. Don't dumb it down**

Complexity should be reduced but not obscured.

lb.com/twig#principles

DRUPAL 8 THEME LAYER

Principals to guide us

- 1. Start with nothing**
- 2. Build from use cases**
- 3. Provide tools**
- 4. Consolidate**
- 5. Visibility**
- 6. Consistency**
- 7. Don't dumb it down**
- 8. Organization should be driven by meaning and semantics over technical convenience**

Consider what an element means rather than how it structurally appears. Theme developers want to see markup in templates, not abstraction.

A NEW THEME ENGINE?



TWIG

[ABOUT](#)[DOCUMENTATION](#)[BLOG](#)[DEVELOPMENT](#)[CONTRIBUTORS](#)

Twig Documentation

Read the online documentation to learn more about Twig.

- [Introduction / Installation](#)
- [Twig for Template Designers](#)
- [Twig for Developers](#)
- [Extending Twig](#)
- [Twig Internals](#)
- [Twig Recipes](#)
- [Coding Standards](#)
- [API](#)
- [Twig contributed extensions documentation](#)
- [License](#)

Twig Reference

Browse the online reference to learn more about built-in features.

You can download the documentation for offline reading:



well documented

TWIG

Extending Twig

Twig can be extended in many ways; you can add extra tags, filters, tests, operators, global variables, and functions. You can even extend the parser itself with node visitors.



The first section of this chapter describes how to extend Twig easily. If you want to reuse your changes in different projects or if you want to share them with others, you should then create an extension as described in the following section.

extensible

TWIG

- **Secure:** When it comes to security, Twig has some unique features:
 - *Automatic output escaping:* To be on the safe side, you can enable automatic output escaping globally or for a block of code:

```
{% autoescape true %}
  {% var %}
  {% var|raw %}      {%# var won't be escaped #}
  {% var|escape %}   {%# var won't be doubled-escaped #}
{% endautoescape %}
```

- *Sandboxing:* Twig can evaluate any template in a sandbox environment where the user has access to a limited set of tags, filters, and object methods defined by the developer. Sandboxing can be enabled globally or locally for just some templates:

```
{% include "user.html" sandboxed %}
```

secure

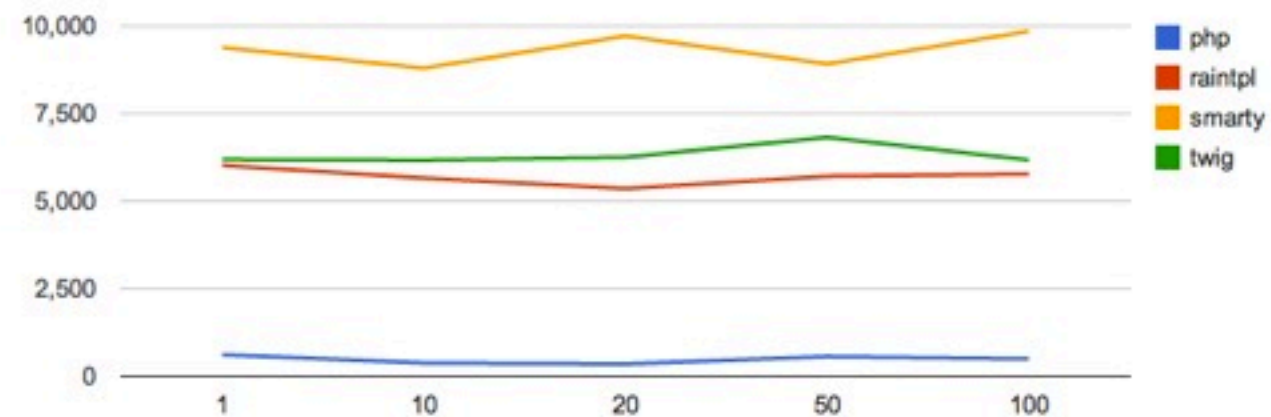
TWIG

Summary (assign)

Test	tot. time	tot. memory	package size
<u>php</u> 5.3.3-7+squeeze9	483 μ s	11.97 KB	4 KB
<u>raintpl</u> 2.7.0	5707 μ s	282.77 KB	37 KB
<u>twig</u> 1.5.1	6323 μ s	715.93 KB	647 KB
<u>smarty</u> 3.1.7	9336 μ s	1.28 MB	971 KB

Execution Time (assign)

Execution Time (μ s)



fast

TWIG



IDE integration

TWIG



recognizable syntax

TWIG



by the creator of Symfony, Fabien Potencier

TWIG

what does it look like?

TWIG

what does it look like?

```
{% if items %}
<div class="item-list"> {% @todo remove this div %}
  {% if title %}
    <h3>{{ title }}</h3>
  {% endif %}
  {% if type == 'ol' %}
    <ol class="{{ attributes.class }}"{{ attributes }}>
  {% else %}
    <ul class="{{ attributes.class }}"{{ attributes }}>
  {% endif %}
  {% for item in items %}
    <li{{ item.attributes }}>{{ item.data }}</li>
  {% endfor %}
  {% if type == 'ol' %}
    </ol>
  {% else %}
    </ul>
  {% endif %}
</div> {% @todo remove this div %}
{% endif %}
```


TWIG

print with {{ }}

```
{% if items %}
<div class="item-list"> {% @todo remove this div %}
    {% if title %}
        <h3>{{ title }}</h3>
    {% endif %}
    {% if type == 'ol' %}
        <ol class="{{ attributes.class }}"{{ attributes }}>
    {% else %}
        <ul class="{{ attributes.class }}"{{ attributes }}>
    {% endif %}
    {% for item in items %}
        <li{{ item.attributes }}>{{ item.data }}</li>
    {% endfor %}
    {% if type == 'ol' %}
        </ol>
    {% else %}
        </ul>
    {% endif %}
</div> {% @todo remove this div %}
{% endif %}
```

TWIG

commands with {% %}

```
{% if items %}
<div class="item-list"> {% @todo remove this div #}
  {% if title %}
    <h3>{{ title }}</h3>
  {% endif %}
  {% if type == 'ol' %}
    <ol class="{{ attributes.class }}"{{ attributes }}>
  {% else %}
    <ul class="{{ attributes.class }}"{{ attributes }}>
  {% endif %}
  {% for item in items %}
    <li class="{{ item.attributes }}">{{ item.data }}</li>
  {% endfor %}
  {% if type == 'ol' %}
    </ol>
  {% else %}
    </ul>
  {% endif %}
</div> {% @todo remove this div #}
{% endif %}
```


TWIG

comments with {# #}

```
{% if items %}
<div class="item-list"> {# @todo remove this div #}
  {% if title %}
    <h3>{{ title }}</h3>
  {% endif %}
  {% if type == 'ol' %}
    <ol class="{{ attributes.class }}"{{ attributes }}>
  {% else %}
    <ul class="{{ attributes.class }}"{{ attributes }}>
  {% endif %}
  {% for item in items %}
    <li>{{ item.attributes }}>{{ item.data }}</li>
  {% endfor %}
  {% if type == 'ol' %}
    </ol>
  {% else %}
    </ul>
  {% endif %}
</div> {# @todo remove this div #}
{% endif %}
```

TWIG

drill down into all variables with `.`

```
{% if items %}
<div class="item-list"> {% @todo remove this div %}
  {% if title %}
    <h3>{{ title }}</h3>
  {% endif %}
  {% if type == 'ol' %}
    <ol class="{{ attributes.class }}"{{ attributes }}>
  {% else %}
    <ul class="{{ attributes.class }}"{{ attributes }}>
  {% endif %}
  {% for item in items %}
    <li{{ item.attributes }}>{{ item.data }}</li>
  {% endfor %}
  {% if type == 'ol' %}
    </ol>
  {% else %}
    </ul>
  {% endif %}
</div> {% @todo remove this div %}
{% endif %}
```


TWIG

All markup is generated in template files.

```
- */
-function theme_
- $items = $var
- $title = (str
- // @todo 'type
- $type = $var
- $list_attribu
-
- $output = '';
- if ($items) {
-     $output .=
-
-     $num_items =
-     $i = 0;
-     foreach ($i
-         $i++;
-         $attribute
-         if (is_ar
-             if (isse
-                 $attr:
-         }
-         $item =
-     }
-     $attribute
-     if ($i ==
-         $attribu
-     }
-     if ($i ==
-         $attribu
-     }
-     $output .=
```

```
--- /dev/null
+++ b/core/themes/stark/templates/system/item-list.twig
@@ -0,0 +1,40 @@
+{#
+/**
+ * @file
+ * Returns HTML for a list or nested list of items.
+ *
+ * Available variables:
+ *
```

D7 PAIN POINTS

does Twig help?

1. Mixed data types
2. Different methods of printing variables
3. Two ways to override markup
4. Too many template files & theme functions
5. Insecure
6. Drupal-specific
7. A complex mix of subsystems

D7 PAIN POINTS

does Twig help?

I. Mixed data types

D7 PAIN POINTS

does Twig help?

I. Mixed data types

**All template variables are accessed
consistently:**

node.nid
content.links

D7 PAIN POINTS

does Twig help?

I. Mixed data types **FIXED**

D7 PAIN POINTS

does Twig help?

1. Mixed data types **FIXED**
2. Different methods of printing variables

D7 PAIN POINTS

does Twig help?

1. Mixed data types **FIXED**
2. Different methods of printing variables

Removed calls to render() from templates:

```
{{ node.nid }}  
{{ content.links }}
```

D7 PAIN POINTS

does Twig help?

1. Mixed data types **FIXED**
2. Different methods of printing variables **FIXED**

D7 PAIN POINTS

does Twig help?

1. Mixed data types **FIXED**
2. Different methods of printing variables **FIXED**
3. Two ways to override markup

D7 PAIN POINTS

does Twig help?

1. Mixed data types **FIXED**
2. Different methods of printing variables **FIXED**
3. Two ways to override markup

All theme functions become templates.

node.tpl.php becomes node.html.twig
theme_table() becomes table.html.twig

D7 PAIN POINTS

does Twig help?

1. Mixed data types **FIXED**
2. Different methods of printing variables **FIXED**
3. Two ways to override markup **FIXED**

D7 PAIN POINTS

does Twig help?

1. Mixed data types **FIXED**
2. Different methods of printing variables **FIXED**
3. Two ways to override markup **FIXED**
4. Too many template files & theme functions

D7 PAIN POINTS

does Twig help?

1. Mixed data types **FIXED**
2. Different methods of printing variables **FIXED**
3. Two ways to override markup **FIXED**
4. Too many template files & theme functions

We're working on this right now

D7 PAIN POINTS

does Twig help?

1. Mixed data types **FIXED**
2. Different methods of printing variables **FIXED**
3. Two ways to override markup **FIXED**
4. Too many template files & theme functions **@todo**

D7 PAIN POINTS

does Twig help?

1. Mixed data types **FIXED**
2. Different methods of printing variables **FIXED**
3. Two ways to override markup **FIXED**
4. Too many template files & theme functions **@todo**
5. Insecure

D7 PAIN POINTS

does Twig help?

1. Mixed data types **FIXED**
2. Different methods of printing variables **FIXED**
3. Two ways to override markup **FIXED**
4. Too many template files & theme functions **@todo**
5. Insecure

**All variables are *automatically* sanitized
and most PHP functions
cannot be executed in template files.**

D7 PAIN POINTS

does Twig help?

1. Mixed data types **FIXED**
2. Different methods of printing variables **FIXED**
3. Two ways to override markup **FIXED**
4. Too many template files & theme functions **@todo**
5. Insecure **FIXED**

D7 PAIN POINTS

does Twig help?

1. Mixed data types **FIXED**
2. Different methods of printing variables **FIXED**
3. Two ways to override markup **FIXED**
4. Too many template files & theme functions **@todo**
5. Insecure **FIXED**
6. Drupal-specific

D7 PAIN POINTS

does Twig help?

1. Mixed data types **FIXED**
2. Different methods of printing variables **FIXED**
3. Two ways to override markup **FIXED**
4. Too many template files & theme functions **@todo**
5. Insecure **FIXED**
6. Drupal-specific

**Twig is used elsewhere on the web
is syntactically similar to other languages
and looks a lot more like HTML.**

D7 PAIN POINTS

does Twig help?

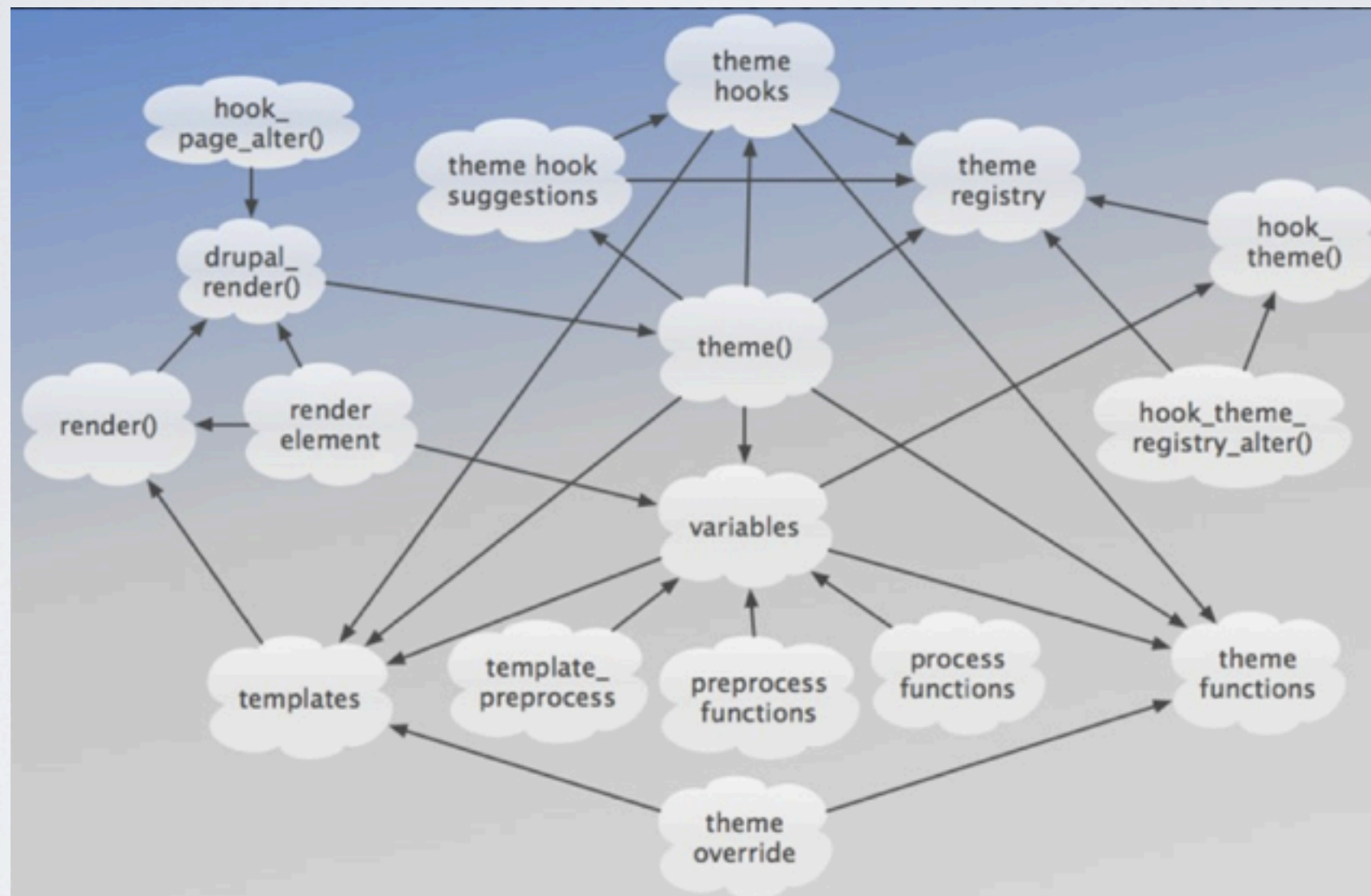
1. Mixed data types **FIXED**
2. Different methods of printing variables **FIXED**
3. Two ways to override markup **FIXED**
4. Too many template files & theme functions **@todo**
5. Insecure **FIXED**
6. Drupal-specific **FIXED**

D7 PAIN POINTS

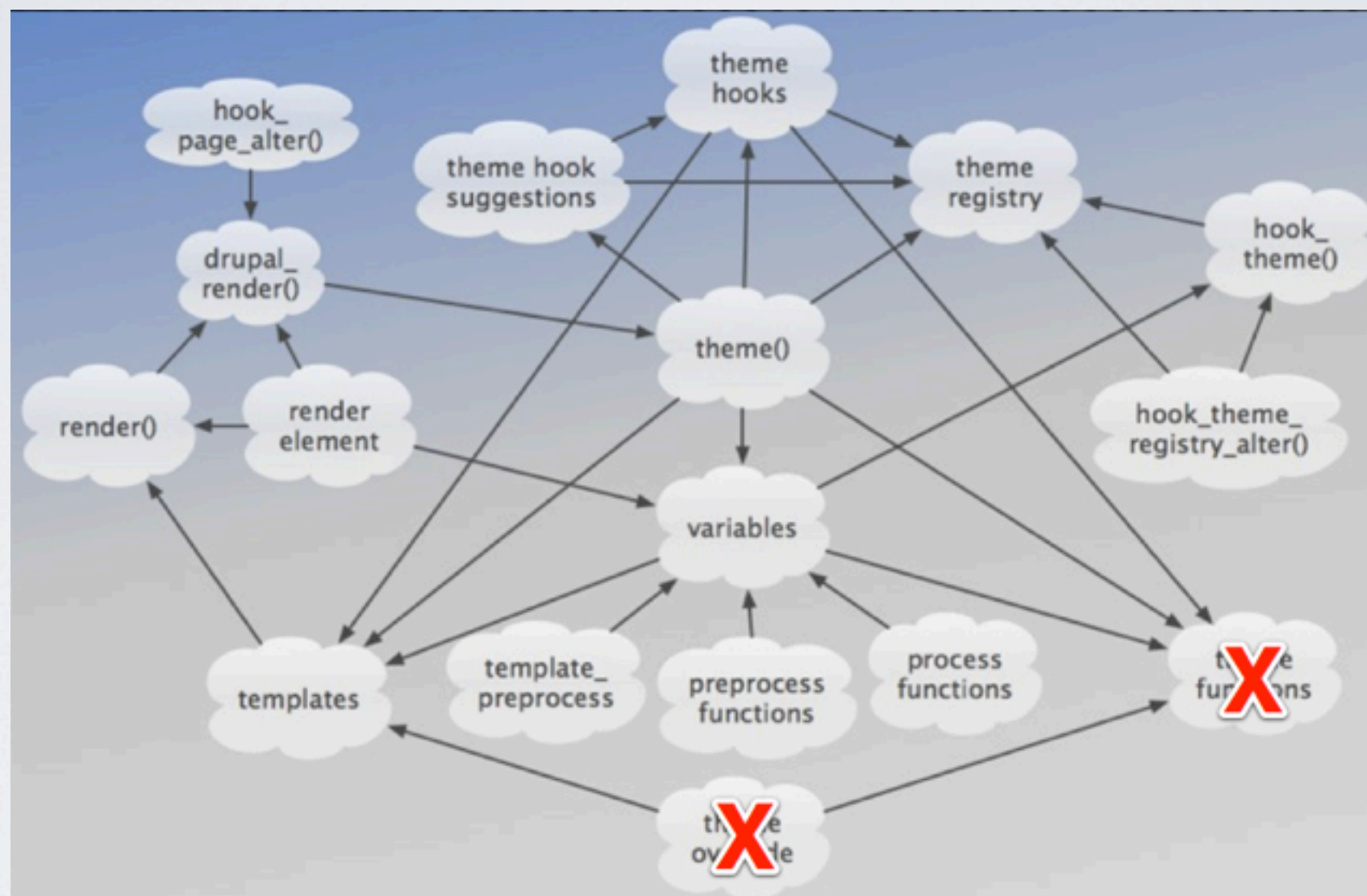
does Twig help?

1. Mixed data types **FIXED**
2. Different methods of printing variables **FIXED**
3. Two ways to override markup **FIXED**
4. Too many template files & theme functions **@todo**
5. Insecure **FIXED**
6. Drupal-specific **FIXED**
7. A complex mix of subsystems

D7 THEME LAYER

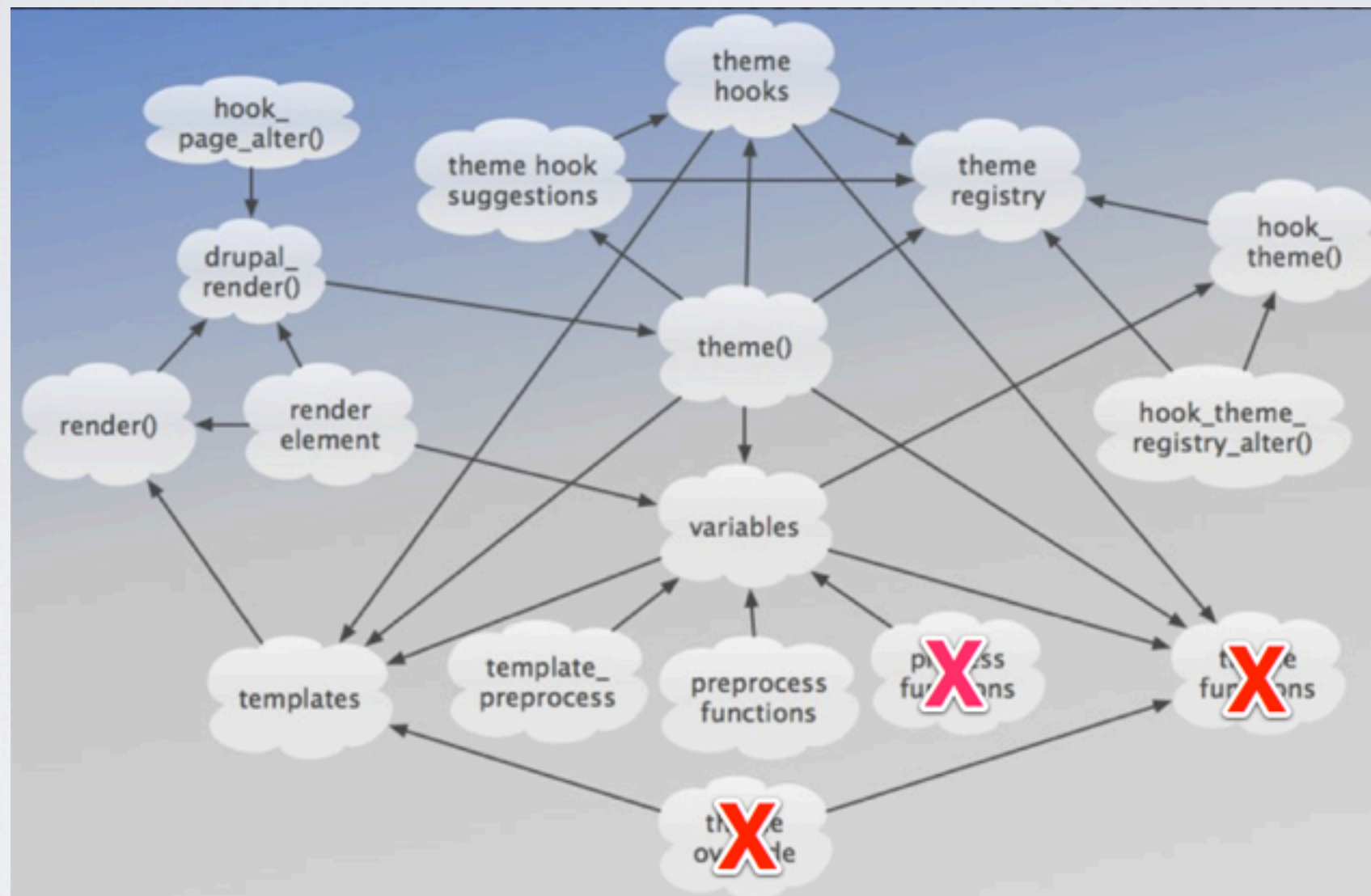


D8 IMPROVEMENTS



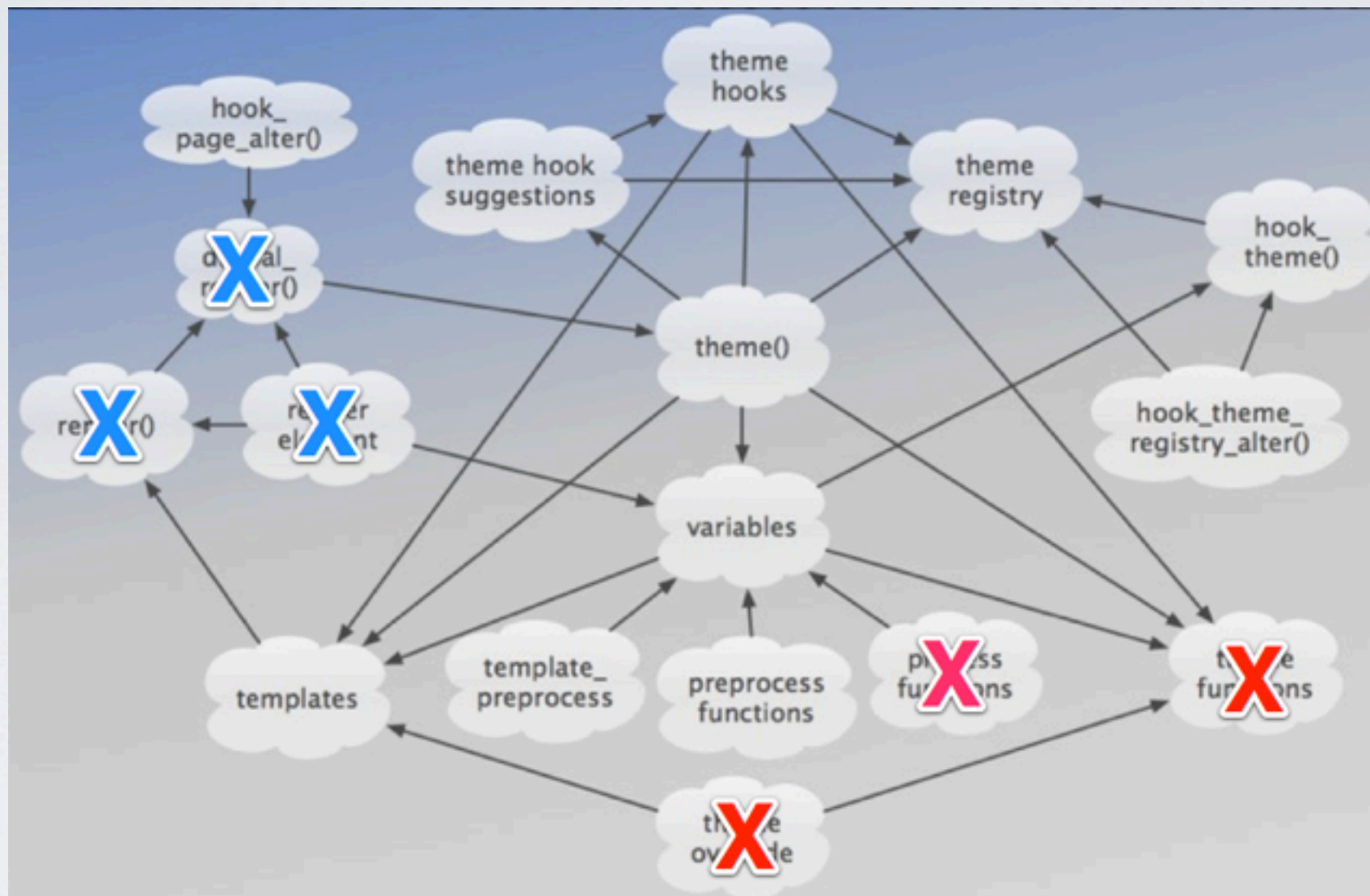
remove theme functions (and overrides) entirely.

D8 IMPROVEMENTS



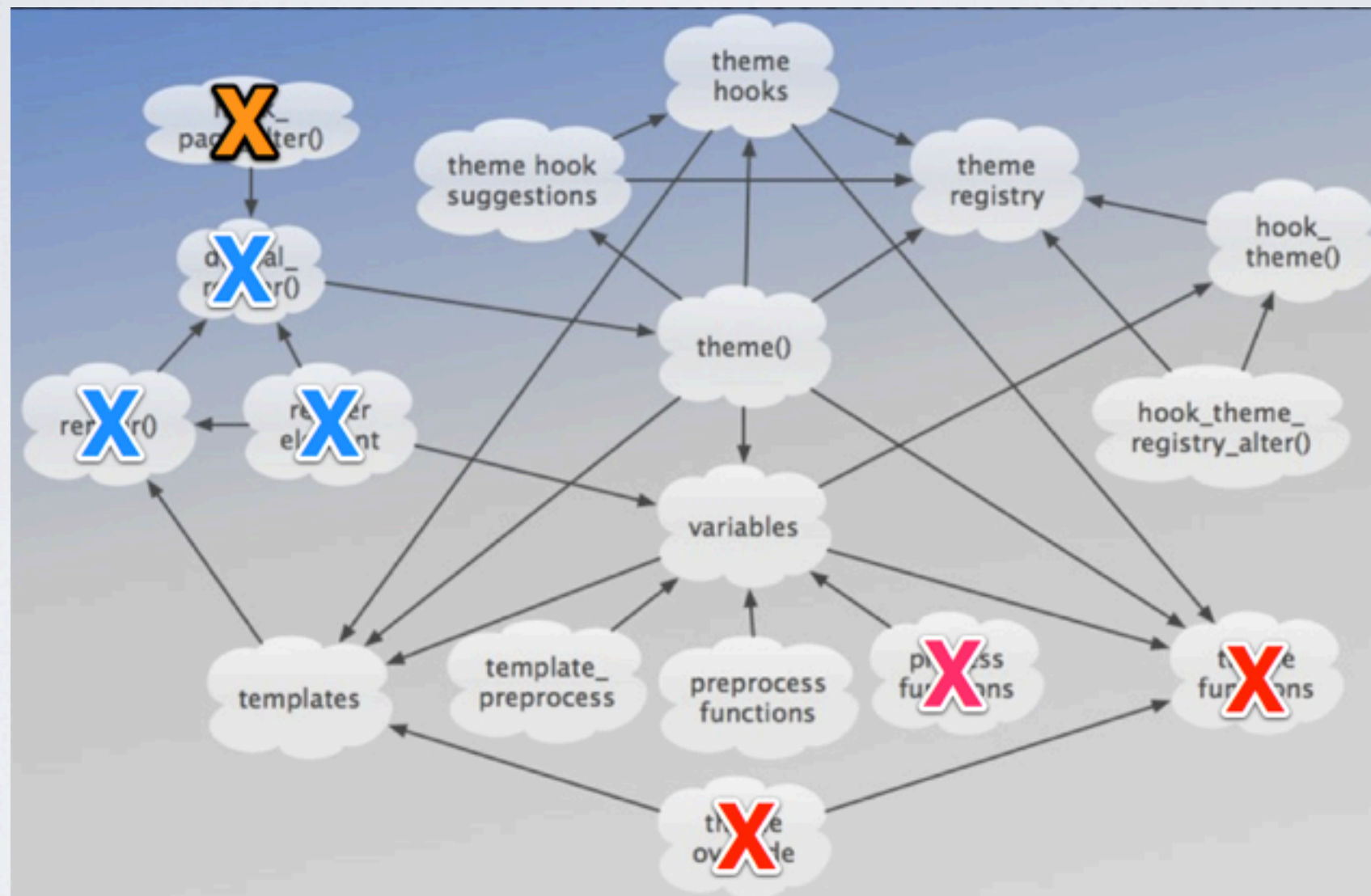
remove process.

D8 IMPROVEMENTS



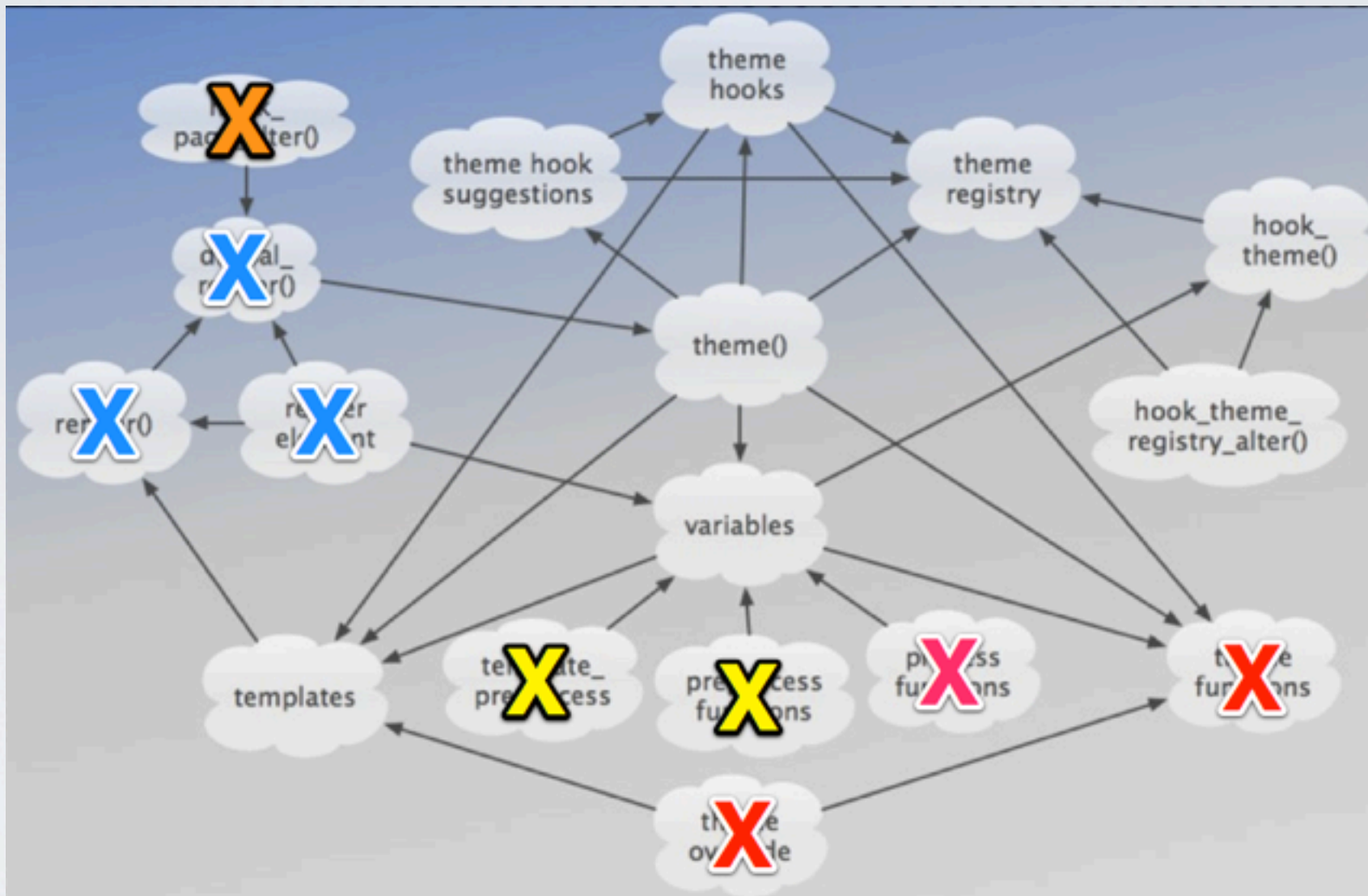
remove render.

D8 IMPROVEMENTS



remove page alter?

D9 IMPROVEMENTS



remove preprocess?

D7 PAIN POINTS

does Twig help?

1. Mixed data types **FIXED**
 2. Different methods of printing variables **FIXED**
 3. Two ways to override markup **FIXED**
 4. Too many template files & theme functions **@todo**
 5. Insecure **FIXED**
 6. Drupal-specific **FIXED**
 7. A complex mix of subsystems
- We can remove all theme functions, render, process & (maybe) preprocess too.**

D7 PAIN POINTS

does Twig help?

1. Mixed data types **FIXED**
2. Different methods of printing variables **FIXED**
3. Two ways to override markup **FIXED**
4. Too many template files & theme functions **@todo**
5. Insecure **FIXED**
6. Drupal-specific **FIXED**
7. A complex mix of subsystems **@todo**

D7 PAIN POINTS

does Twig help?

D7 PAIN POINTS

does Twig help?

YES!!!

TWIG: OTHER WINS

TWIG: OTHER WINS

less code than PHP functions

TWIG: OTHER WINS

less code than PHP functions

```
function theme_image($variables) {  
    $attributes = $variables['attributes'];  
    $attributes['src'] = file_create_url($variables['uri']);  
  
    foreach (array('width', 'height', 'alt', 'title') as $key) {  
        if (isset($variables[$key])) {  
            $attributes[$key] = $variables[$key];  
        }  
    }  
  
    return '<img' . new Attribute($attributes) . ' />';  
}
```

D7

```

```

D8

theme_image becomes image.html.twig

TWIG

less code than PHP functions

D7

```
function theme_username($variables) {  
    if (isset($variables['link'])) {  
        $output = $variables['link'];  
    }  
    else {  
        $output = '<span' . new Attribute($variables['attributes']) . '>' . $variables['name'] . $variables['extra'] . '</span>';  
    }  
    return $output;  
}
```

D8

```
{% if link %}  
    <a href="{{ link.href }}" {{ link.attributes }}>{{ name }}{{ extra }}</a>  
{% else %}  
    <span{{ attributes }}>{{ name }}{{ extra }}</span>  
{% endif %}
```

theme_username becomes username.html.twig

TWIG

less code than PHP functions

```
function theme_link($variables) {  
    return '<a href="' . check_plain(url($variables['path'], $variables['options']))  
        . '"' . new Attribute($variables['options']['attributes']) . '>  
        . ($variables['options']['html'] ? $variables['text'] : check_plain($variables['text']))  
        . '</a>';  
}
```

D7

D8

```
<a href="{{ url(path, options) }}"|{{ attributes }}>{{ text }}</a>
```

theme_link becomes link.html.twig

D7

```

function theme_item_list($variables) {
  $items = $variables['items'];
  $title = $variables['title'];
  $type = $variables['type'];
  $list_attributes = $variables['attributes'];

  $output = '';
  if ($items) {
    $output .= '<' . $type . drupal_attributes($list_attributes) . '>';

    $num_items = count($items);
    $i = 0;
    foreach ($items as $key => $item) {
      $i++;
      $attributes = array();

      if (is_array($item)) {
        $value = '';
        if (isset($item['data'])) {
          $value = $item['data'];
        }
        $attributes = array_diff_key($item, array('data' => 0, 'children' => 0));

        // Append nested child list, if any.
        if (isset($item['children'])) {
          // HTML attributes for the outer list are defined in the 'attributes'
          // theme variable, but not inherited by children. For nested lists,
          // all non-numeric keys in 'children' are used as list attributes.
          $child_list_attributes = array();
          foreach ($item['children'] as $child_key => $child_item) {
            if (is_string($child_key)) {
              $child_list_attributes[$child_key] = $child_item;
              unset($item['children'][$child_key]);
            }
          }
          $value .= theme('item_list', array(
            'items' => $item['children'],
            'type' => $type,
            'attributes' => $child_list_attributes,
          ));
        }
        else {
          $value = $item;
        }

        $attributes['class'][] = ($i % 2 ? 'odd' : 'even');
        if ($i == 1) {
          $attributes['class'][] = 'first';
        }
        if ($i == $num_items) {
          $attributes['class'][] = 'last';
        }

        $output .= '<li' . drupal_attributes($attributes) . '>' . $value . '</li>';
      }
      $output .= "</$type>";
    }

    // Only output the list container and title, if there are any list items.
    if ($output != '') {
      if ($title != '') {
        $title = '<h3>' . $title . '</h3>';
      }
      $output = '<div class="item-list">' . $title . $output . '</div>';
    }

    return $output;
  }
}

```

lots less code than PHP functions

D8

```

{% if items %}
<div class="item-list"> {% @todo remove this div %}
  {% if title %}
    <h3>{{ title }}</h3>
  {% endif %}
  {% if type == 'ol' %}
    <ol class="{{ attributes.class }}"{{ attributes }}>
  {% else %}
    <ul class="{{ attributes.class }}"{{ attributes }}>
  {% endif %}
  {% for item in items %}
    <li{{ item.attributes }}>{{ item.data }}</li>
  {% endfor %}
  {% if type == 'ol' %}
    </ol>
  {% else %}
    </ul>
  {% endif %}
</div> {% @todo remove this div %}
{% endif %}

```

theme_item_list becomes
item_list.html.twig

TWIG: OTHER WINS

Awesome template inspection.

TWIG: OTHER WINS

Awesome template inspection.

```
$settings['twig_debug'] = TRUE;
```

```
218 <!-- THEME DEBUG -->
219 <!-- CALL: theme('item_list__user__new') -->
220 <!-- BEGIN OUTPUT from 'core/modules/system/templates/item-list.html.twig' -->
221 <div class="item-list">
222     <ul class="item-list">
223         <li class="odd first last"><a href="/user/1" title="View user profile." class="username">
224     </ul>
225 </div>
226
227 <!-- END OUTPUT from 'core/modules/system/templates/item-list.html.twig' -->
228
```

(“devel themer” in core)

TWIG: OTHER WINS

Awesome variable inspection.

TWIG: OTHER WINS

Awesome variable inspection.

```
{% dump(_context) %}
```

(devel's dpm() in core)

TWIG: OTHER WINS

Template inheritance

TWIG: OTHER WINS

Template inheritance

```
<section id="comments" class="{{ attributes.class }}"{{ attributes }}>
  {% if comments %}
    {% if node.type != 'forum' %}
      {{ title.prefix }}
      <h2 class="title">{{ 'Comments' | t }}</h2>
      {{ title.suffix }}
    {% endif %}

    {{ comments }}
  {% endif %}

  {% if form %}
    <h2 class="title comment-form">{{ 'Add new comment' | t }}</h2>
    {{ form }}
  {% endif %}
</section>
```

comment-wrapper.html.twig before

TWIG: OTHER WINS

Template inheritance

```
<section id="comments" class="{{ attributes.class }}"{{ attributes }}>
  {% if comments %}
    {% codeblock title %}
      {{ title.prefix }}
      <h2 class="title">{{ 'Comments' | t }}</h2>
      {{ title.suffix }}
    {% end codeblock %}
    {{ comments }}
  {% endif %}

  {% if form %}
    <h2 class="title comment-form">{{ 'Add new comment' | t }}</h2>
    {{ form }}
  {% endif %}
</section>
```

comment-wrapper.html.twig after

TWIG: OTHER WINS

Template inheritance

```
{% extends "comment-wrapper.html.twig" %}~  
  
{% codeblock title %}~  
{% end codeblock %}~
```

comment-wrapper--forum.html.twig
(child template)

TWIG: OTHER WINS

Possible performance gains (Much TBD)

TWIG: OTHER WINS

Possible performance gains (Much TBD)

- PHPtemplate reads files from disk **on every use** (or stat()s them with APC)
- Twig templates are read once & compiled into classes

TWIG: OTHER WINS

Possible performance gains (Much TBD)

- PHPtemplate reads files from disk **on every use** (or stat()s them with APC)
- Twig templates are read once & compiled into classes

Rendering should get much faster when the same content element appears multiple times on the page.

TWIG: OTHER WINS

Possible performance gains (Much TBD)

- PHPtemplate reads files from disk **on every use** (or stat()s them with APC)
- Twig templates are read once & compiled into classes

Rendering should get much faster when the same content element appears multiple times on the page.

Consolidating many similar templates will result in an additional gain.

TWIG: OTHER WINS

In-browser template editing finally safe.

TWIG: OTHER WINS

In-browser template editing finally safe.

- Saving PHP code in the database is a HUGE no-no.
- Twig is not PHP, and is safe to store!

TWIG: OTHER WINS

In-browser template editing finally safe.

- Saving PHP code in the database is a HUGE no-no.
- Twig is not PHP, and is safe to store!

Modules like 'Contemplate' (Content templates) will finally be safe to use.

TWIG: OTHER WINS

In-browser template editing finally safe.

- Saving PHP code in the database is a HUGE no-no.
- Twig is not PHP, and is safe to store!

Modules like 'Contemplate' (Content templates) will finally be safe to use.

In-browser template editing is something WordPress users have been asking of Drupal for a very long time.

TWIG: OTHER WINS

Twig template files can be used on the front end, too.

TWIG: OTHER WINS

Twig template files can be used on the front end, too.

One template can return markup for both your PHP-generated pages, as well as pages generated vi JS in AJAX callbacks.

TWIG: OTHER WINS

Twig template files can be used on the front end, too.

One template can return markup for both your PHP-generated pages, as well as pages generated vi JS in AJAX callbacks.

We can use other open source libraries like TwigJS.
(<https://github.com/schmittjoh/twig.js>)

TWIG: OTHER WINS

2-way communication between UI and code.

TWIG: OTHER WINS

2-way communication between UI and code.

No broken UIs.

TWIG: OTHER WINS

2-way communication between UI and code.

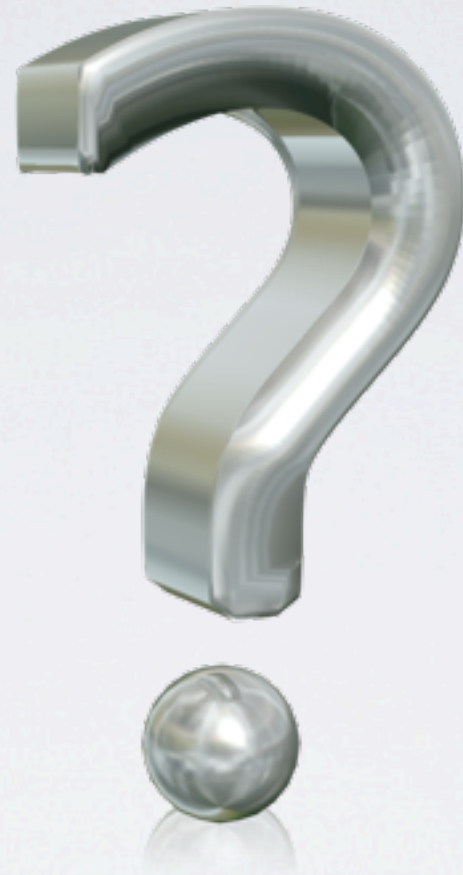
No broken UIs.

Template files can be created first, and the Drupal site can build itself* based on the presence and location of variables in the templates.

*within reason

THE NEW THEME LAYER IN DRUPAL 8

looks pretty awesome, right?



QUESTIONS?

HELP US GET THERE

we need your help!

we could be “done” in 1.5 weeks

<http://drupal.org/node/1757550>

IRC: #drupal-twig

THE NEW THEME LAYER IN DRUPAL 8

Jen Lampton ~ @jenlampton
www.jenlampton.com

photo credits:

lolcat-flexible

<http://cheezburger.com/2679924736>

anything is possible pebbles

<http://www.invergordontours.com/aip.html>

lolcat questionmark

<http://icanhascheezburger.com/2007/10/31/11197/>

wheel-reinvented

<http://www.brainwads.net/drewhawkins/2012/01/dont-re-invent-the-wheel-make-something-better/>

objects

http://2teachers1classroom.blogspot.com/2009_02_01_archive.html

shapes

http://englishclass.jp/reading/topic/For_Screening_Purposes_Only

secure

<http://blog.stratopedia.org/2010/06/03/what-is-a-secure-site/>

consistency

<http://icsigns.org/press/2010/03/23/consistency-staying-on-the-mark/>

twig bird comic

<http://s302.photobucket.com/albums/nn105/walkseva/?action=view¤t=thebirdneedsthattwig.gif¤ttag=bird%20park%20twig%20comic%20need%20it>

twig docs screenshots

<http://twig.sensiolabs.org/documentation>

twig speed graphs

<http://phpcomparison.net/>

python icon

<http://python-hosting.org/>

ruby icon

<http://itmediaconnect.ro/en/web>

django logo

<http://py-arahat.blogspot.com/2010/08/django-vs-pylons.html>

symfony logo

<http://symfony.com/logo>

scotch glass

<http://www.thespir.it/articles/scotch-101/?viewall=1>

speech bubble

<http://lutfash.blogspot.com/2012/10/youre-beautiful-no-matter-what-they-say.html>