

Drupal 7 Render API

Slides available at:
fs.io/files/d7camp.pdf

Presented by:



FreestyleSystems

Richard Burford
Twitter: @psynaptic
IRC: psynaptic
drupal.org: psynaptic



COLUGO

INTERNET SOLUTIONS

Simon Poulston
Twitter: @soulston
IRC: soulston
drupal.org: soulston

Theming in Drupal 7 Kicks A\$\$

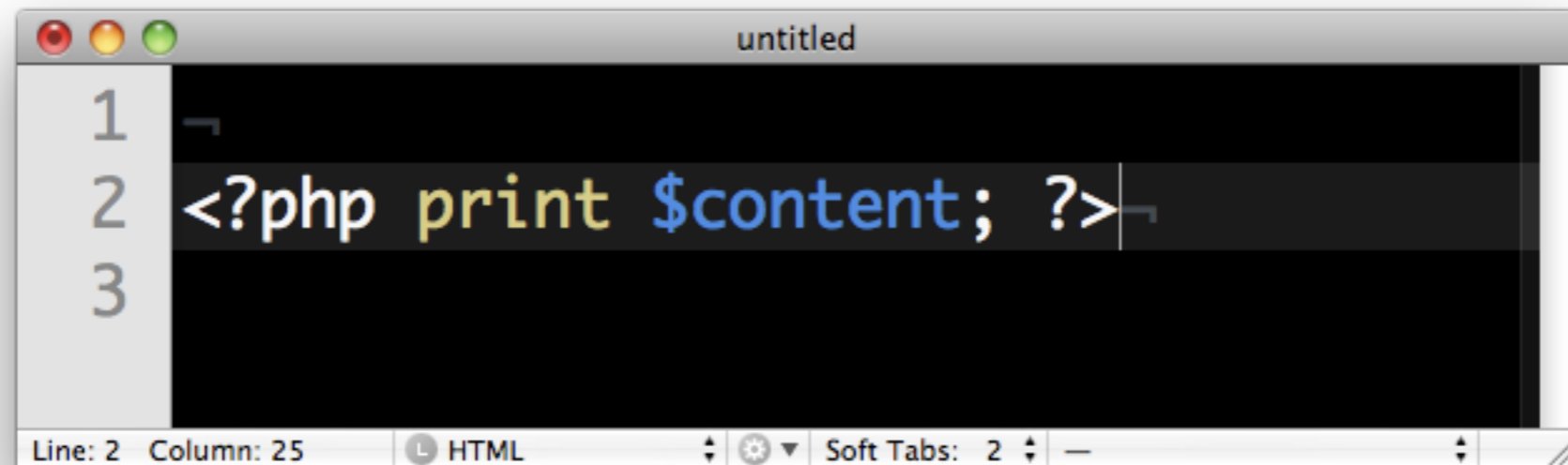


Drupal 6

The biggest change in the theme layer in D7 is that everything is now a render array instead of a string, for example:

D6 template file:

The content variable is a huge string of pre-rendered HTML markup. This sucks!!!



```
1  
2 <?php print $content; ?>  
3
```

The screenshot shows a code editor window with a dark background. The title bar reads 'untitled'. On the left, there is a line number column with '1', '2', and '3'. The main text area contains the PHP code: `<?php print $content; ?>`. The status bar at the bottom indicates 'Line: 2 Column: 25', 'HTML', and 'Soft Tabs: 2'.

Why is this bad?

Why is this bad?



Why is this bad?

To alter anything in this string you pretty much have to use regex, which sucks.

It was very common in D6 to delete the \$content variable from the template and print fields individually.

This means that if you add a new module that generates a content element, it will not display until you print the element in the template file.

Drupal 7

How has this changed in Drupal 7?

D7 template file:

The content variable is a structured array of content elements.

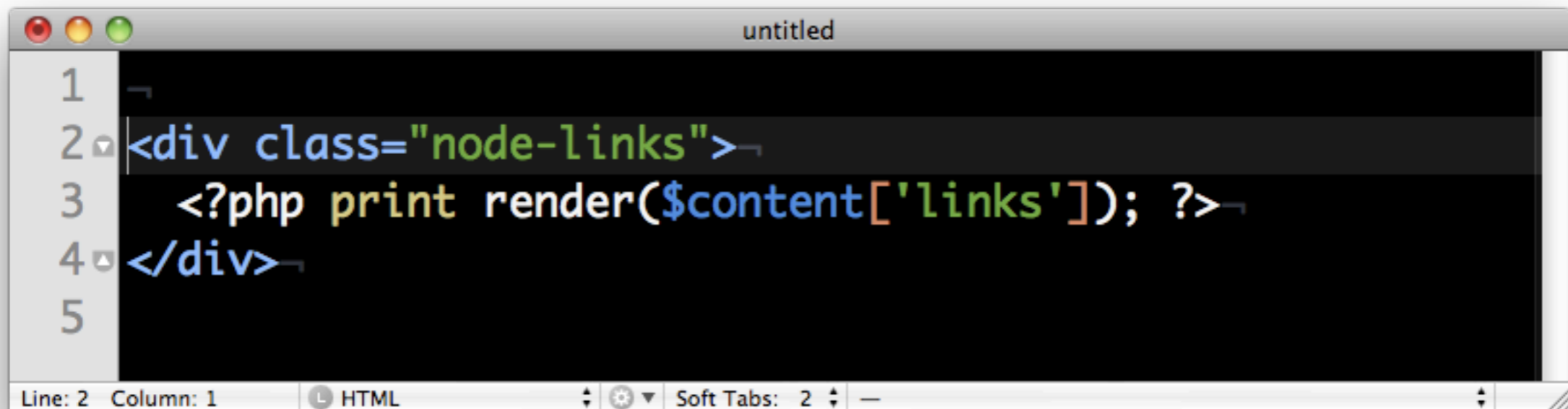


```
1  
2 <?php print render($content); ?>  
3
```

Line: 1 Column: 1 HTML Soft Tabs: 2

Printing individual items

Individual items can be printed from the content array when required:



```
1  
2 <div class="node-links">  
3   <?php print render($content['links']); ?>  
4 </div>  
5
```

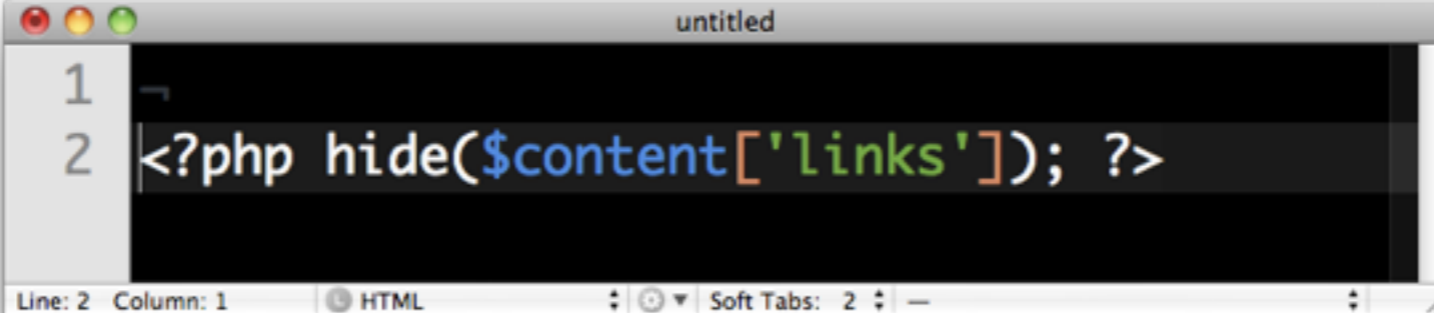
The screenshot shows a code editor window with a dark background and light text. The window title is "untitled". The code is as follows:

```
1  
2 <div class="node-links">  
3   <?php print render($content['links']); ?>  
4 </div>  
5
```

The status bar at the bottom indicates "Line: 2 Column: 1", "HTML", and "Soft Tabs: 2".

Hiding items

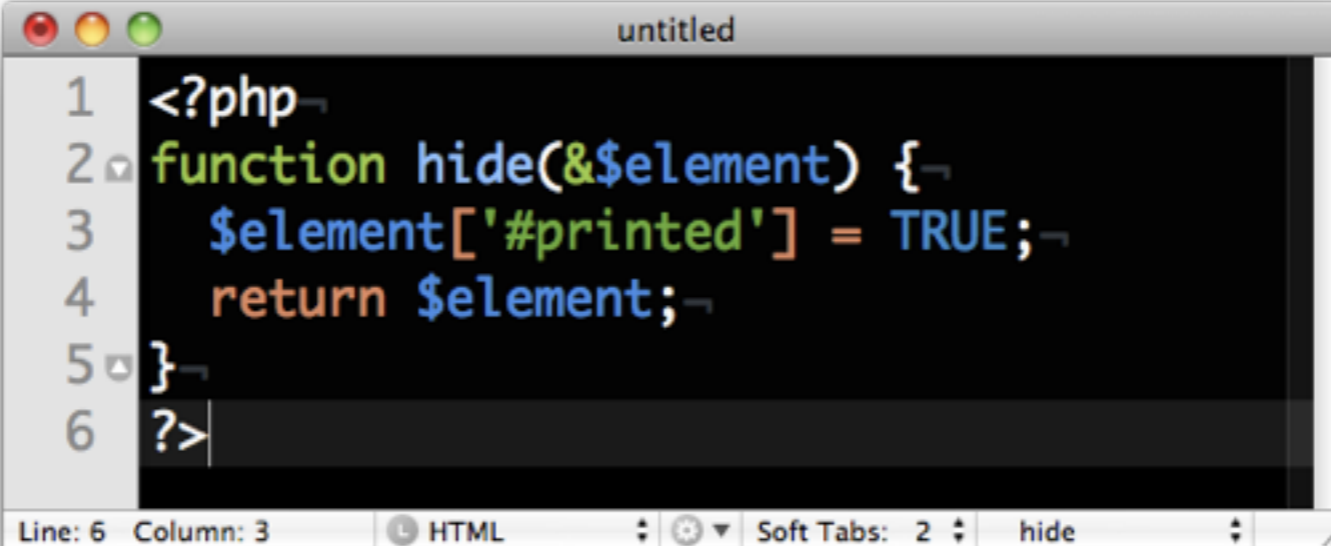
Items can be hidden:



```
1  
2 <?php hide($content['links']); ?>
```

The screenshot shows a code editor window titled 'untitled'. The code is written in PHP and uses the `hide()` function. The status bar at the bottom indicates 'Line: 2 Column: 1', 'HTML', and 'Soft Tabs: 2'.

The **hide()** function sets the **#printed** property of the element array to **TRUE**:



```
1 <?php  
2 function hide(&$element) {  
3     $element['#printed'] = TRUE;  
4     return $element;  
5 }  
6 ?>
```

The screenshot shows a code editor window titled 'untitled' with the definition of the `hide()` function. The function takes a reference to an element array and sets its `#printed` property to `TRUE`. The status bar at the bottom indicates 'Line: 6 Column: 3', 'HTML', and 'Soft Tabs: 2'.

When **render()** iterates over the array it checks each level to see if it has already been printed, if **#printed** is set to **TRUE** it does not render it again.

When an item is rendered it also has the **#printed** property set to **TRUE**.

Showing items

The counterpart to **hide()** is **show()**:



```
1  
2 <?php  
3 function show(&$element) {  
4     $element['#printed'] = FALSE;  
5     return $element;  
6 }  
7 ?>  
8
```

Line: 2 Column: 1 HTML Soft Tabs: 2

RDF

It's worth mentioning that if you're using RDF module to output sub-document-level meta data, you really want to allow as much of core through to the output as possible.

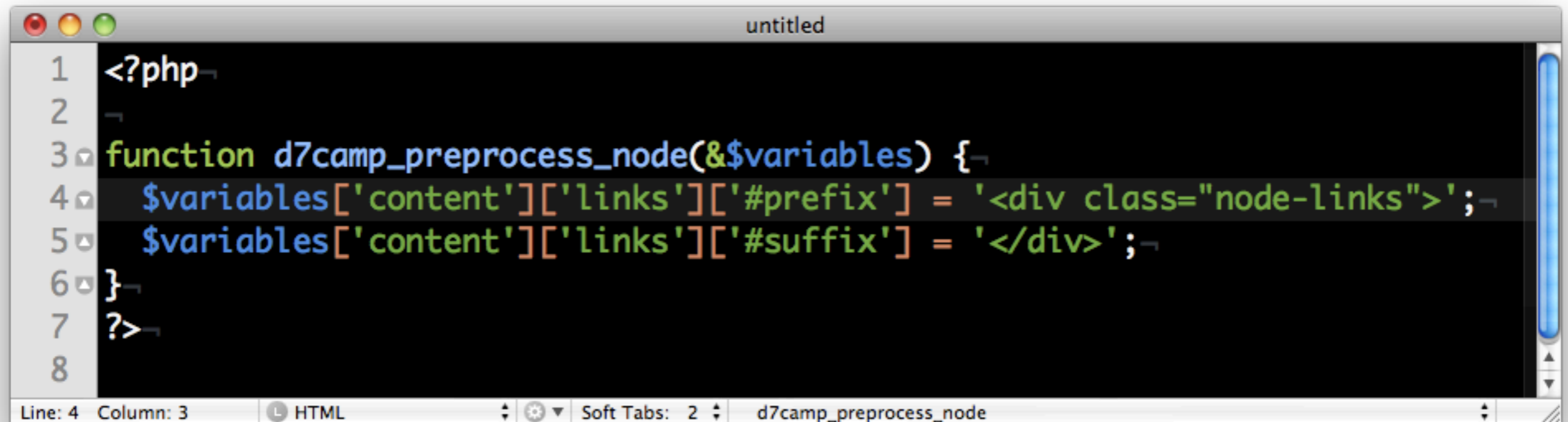
It's good practice to adjust the content being output, rather than rewriting the output.

This also makes the site more maintainable in the future.

Otherwise, you have to rewrite all of your custom code each time you upgrade Drupal or when there are other upstream changes that affect your code.

Preprocessing \$variables

Rather than adjusting the render arrays in the template file, you should do this in preprocess or earlier e.g., **hook_node_view**, **hook_block_view_alter**, and so on:

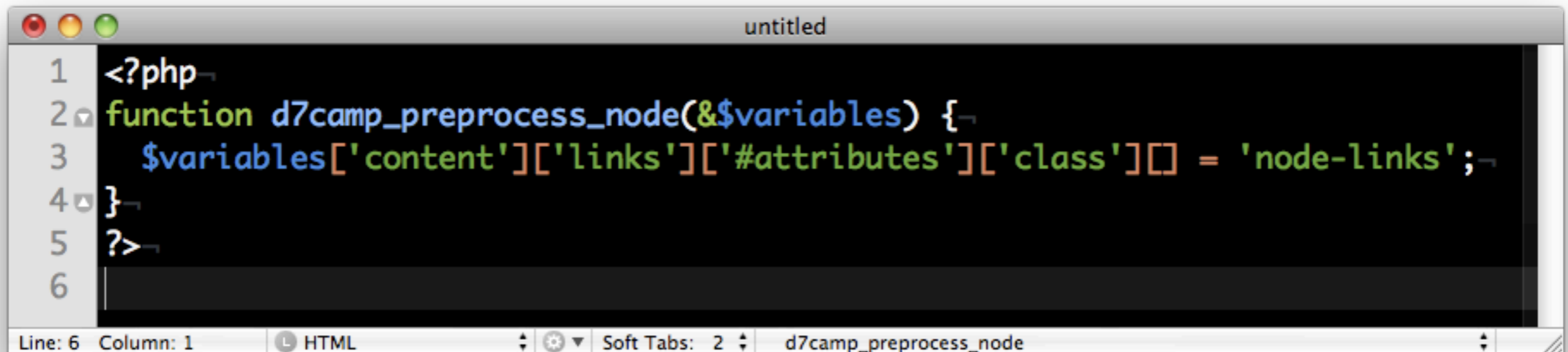


```
1 <?php
2
3 function d7camp_preprocess_node(&$variables) {
4     $variables['content']['links']['#prefix'] = '<div class="node-links">';
5     $variables['content']['links']['#suffix'] = '</div>';
6 }
7 ?>
8
```

Line: 4 Column: 3 HTML Soft Tabs: 2 d7camp_preprocess_node

Adding classes in preprocess

If you need to add an extra class to the links array, you can do this in preprocess:




```
1 <?php
2 function d7camp_preprocess_node(&$variables) {
3     $variables['content']['links']['#attributes']['class'][] = 'node-links';
4 }
5 ?>
6
```

Line: 6 Column: 1 HTML Soft Tabs: 2 d7camp_preprocess_node

Setting weights and other items

You can also set the weight of the element so it appears in the desired order:



```
1  
2 <?php  
3 function d7camp_preprocess_node(&$variables) {  
4     $variables['content']['links']['#prefix'] = '<div class="node-links">';  
5     $variables['content']['links']['#suffix'] = '</div>';  
6     $variables['content']['links']['#weight'] = 7;  
7 }  
8 ?>
```

Line: 2 Column: 1 HTML Soft Tabs: 2

Theme hooks

The **#theme** key for any level of the array defines which theme function is used to render the array.

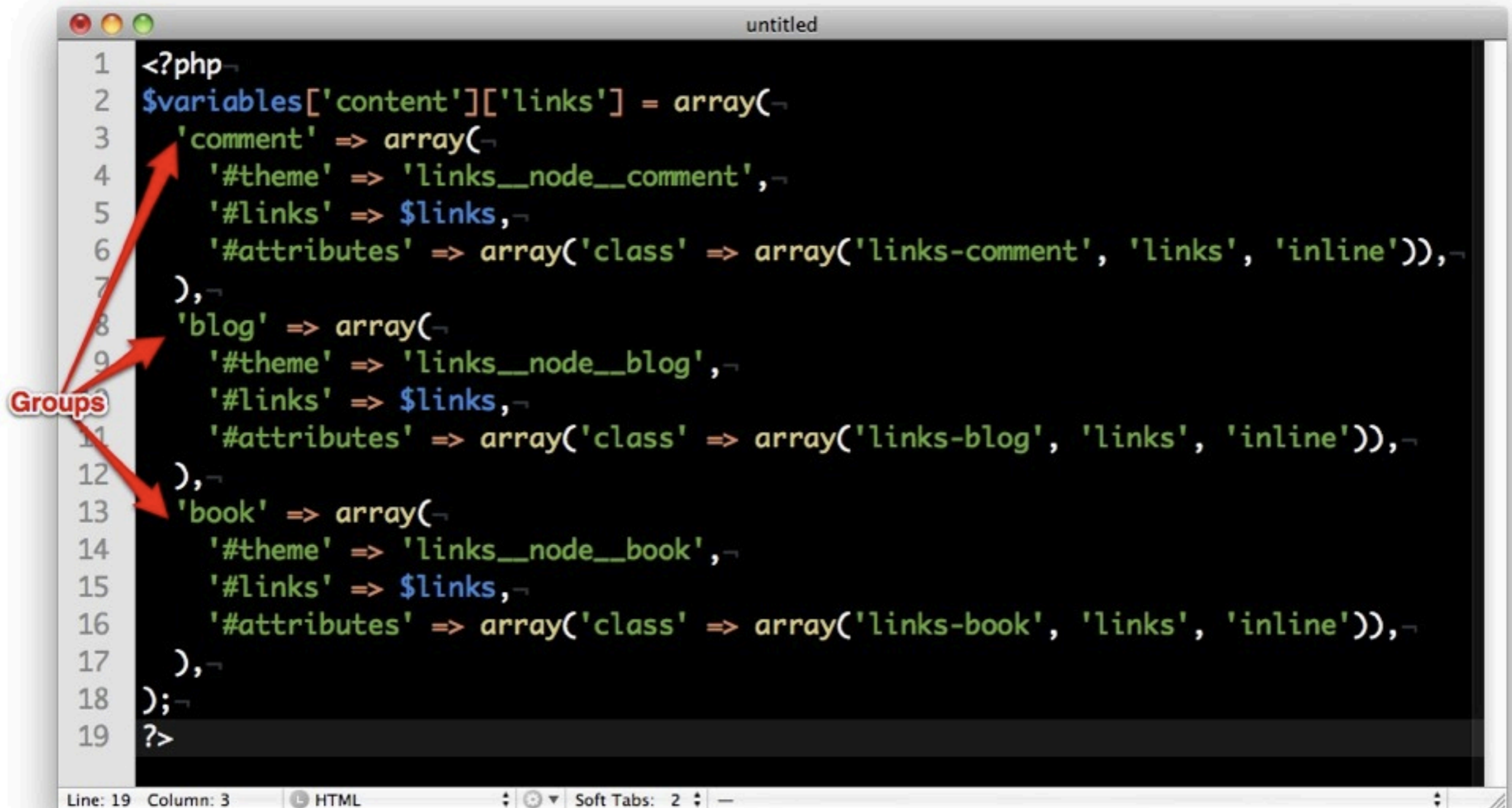


```
1 <?php
2 $content['links'] = array(
3     '#theme' => 'links', // Rendered using theme_links().
4     '#links' => $links,
5     '#attributes' => array('class' => array('links', 'inline')),
6 );
7 ?>
```

Line: 7 Column: 3 HTML Soft Tabs: 2

Group links by type

Links can now *optionally* be **grouped** by the **type** of link, this could be the module that generates the link or some other grouping you wish to apply.



```
1 <?php
2 $variables['content']['links'] = array(
3     'comment' => array(
4         '#theme' => 'links__node__comment',
5         '#links' => $links,
6         '#attributes' => array('class' => array('links-comment', 'links', 'inline')),
7     ),
8     'blog' => array(
9         '#theme' => 'links__node__blog',
10        '#links' => $links,
11        '#attributes' => array('class' => array('links-blog', 'links', 'inline')),
12    ),
13    'book' => array(
14        '#theme' => 'links__node__book',
15        '#links' => $links,
16        '#attributes' => array('class' => array('links-book', 'links', 'inline')),
17    ),
18 );
19 ?>
```

Line: 19 Column: 3 HTML Soft Tabs: 2

Rendering links in the template

You can render these separately in the template:



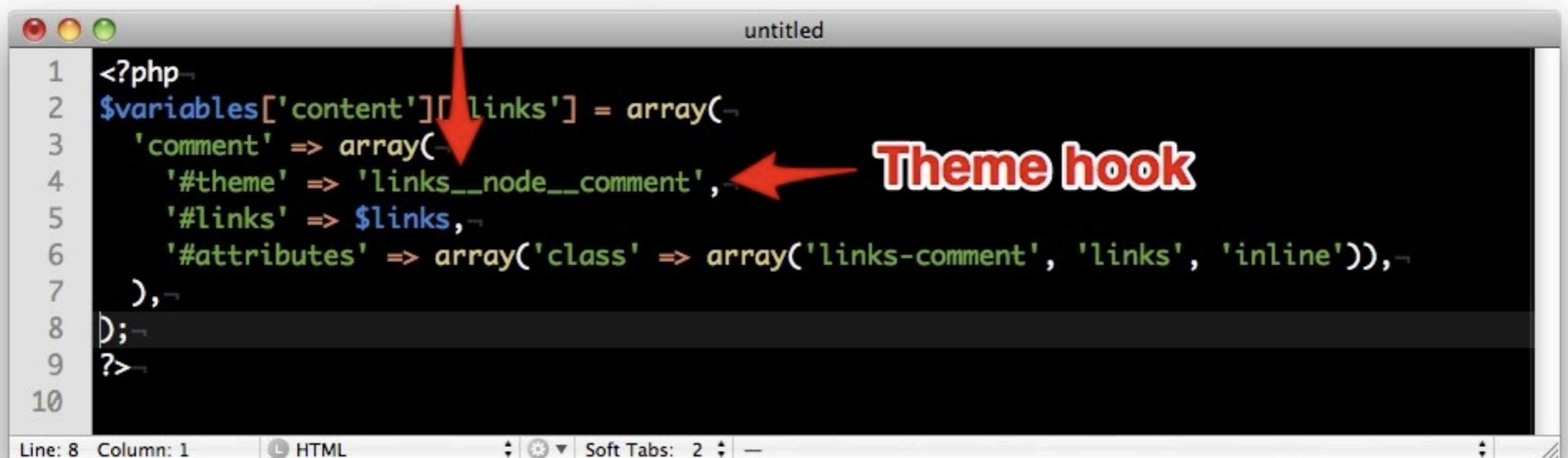
```
1 |  
2 | <div class="comment-links">  
3 |     <?php print render($content['links']['comment']); ?>  
4 | </div>  
5 |  
6 | <div class="blog-links">  
7 |     <?php print render($content['links']['blog']); ?>  
8 | </div>  
9 |  
10 | <?php  
11 |     print render($content); // Also renders the book links.  
12 | ?>  
13 |
```

Line: 1 Column: 1 HTML Soft Tabs: 2

Theme patterns

The **theme hook** named previously use an optional theme "**pattern**" - using the double underscore syntax.

Theme hook "pattern" syntax



```
1 <?php
2 $variables['content']['links'] = array(
3   'comment' => array(
4     '#theme' => 'links__node__comment',
5     '#links' => $links,
6     '#attributes' => array('class' => array('links-comment', 'links', 'inline')),
7   ),
8 );
9 ?>
```

The screenshot shows a code editor window titled "untitled" with PHP code. A red arrow points from the text "Theme hook 'pattern' syntax" to the 'links' key in the array on line 2. Another red arrow points from the text "Theme hook" to the '#theme' key in the array on line 4. The code defines an array structure for rendering links, with a specific theme hook name 'links__node__comment' assigned to the 'comment' element.

If this array is rendered individually (rather than part of a larger array), specific theme functions can be used to render it. This is the order they will be used for the above example: `theme_links__node__comment()`, `theme_links__node()`, `theme_links()`.

Summary

Content is sent to the template file as render arrays

You can render() individual items from the array where you want them

You can use hide() and show() to control which items get rendered in the \$content array

You can easily add classes and modify render arrays in preprocess or earlier

Node links can be grouped, allowing groups to be rendered separately from the rest

Theme hook "patterns" can be used to specify specific theme functions to render arrays

You should now be
Drupal 7 **Render API** ninjas



Questions?

Resources

Render Arrays in Drupal 7

<http://drupal.org/node/930760>

Theme API

<http://drupal.org/node/722174>

Updating themes from D6 to D7

<http://drupal.org/update/themes/6/7>

Drupal Contrib API

<http://drupalcontrib.org>