Drupal Forms API

Adrian Rossouw

Current API (static)

- Functions wrapping HTML markup
- Not really mutable
- Parameter order is hard to remember
- Strong validation is a lot of work

New API (dynamic)

- Elements defined as component arrays
- Similar to menu system
- Elements can be nested
- Elements are weighted
- Drastically reduces amount of functions
- Easier to remember properties

bryght

Static versus Dynamic

- Static forms
  - not easily modifiable
  - requires knowledge of regular expressions
  - fields can be added only pre-form or post-form
  - like raster images
- Dynamic forms
  - stricter adherence to theme layer
  - stricter validation
  - easily modified
  - weighted elements
  - like vector images

Easier to theme

- A theme function for every form (optional)
- Default form renderer that works for most drupal forms
- A simple API to theme forms (one function to learn)
- Can modify properties and call the default renderer
- Classes can be added to elements
- Separation of form definition and form presentation enforced

Easier to validate

- All input elements have a 'valid' property
  - » Can have multiple validations
- All forms have a form wide validation function
- All forms can consult an additional callback (for CCK)

bryght

Easier to modify

- Incredible power at the theme layer
- Ability to ‘hook in’ to any form, and adjust / add / remove elements
- print_r function displays the entire form tree transparently
- Weights allow you to re-order any elements how you want them

bryght

Easier to extend

- New element types
- AOP access to all forms
- Additional properties
  - » Interface levels
  - » Content Construction Kit™
  - » Dynamic text editors
- Add or remove elements from forms

Node form example

```
function node_form($edit) {
   $form['title'] = array(
     type      => 'textfield',
     title     => t('Title'),
     default   => $node->title,
     size      => 60,
     maxlength => 128,
     required  => TRUE
   );
   $form['author'] = array(
     type         => 'fieldset',
     title        => t('Authoring information'),
     collapsible  => TRUE,
     collapsed    => TRUE
   );
   $form['author']['name'] = array(
     type         => 'textfield',
     title        => t('Authored by'),
     default      => $node->name,
     maxlength    => 60,
     autocomplete_path' => 'user/autocomplete'
  );
```

Node form continued

```
    ...

    if ($_POST && form_validate('node_form', $form, $edit) ) {
      form_execute('node_form', $form, $edit);
      drupal_goto('some/place/new');
    }

    return form('node_form', $form, $edit);
  }

  function node_form_validate($form_name, $form, $edit) {
    if ($edit['title'] > $edit['body']) {
      form_set_error($form['title'],
         t('Title can not be longer than the body'));
      return FALSE;
    }
  }

  function node_form_execute($form_name, $form, $edit) {
    /* do whatever you need to do here */
  }
```

Theme node form

```
function theme_node_form($form) {
  $output .= '<div class="node-form">';
    $output .= '<div class="admin">';
      $output .= '<div class="authored">';
      $output .= form_render($form['author']);
      $output .= '</div>';

      $output .= '<div class="options">';
      $output .= form_render($form['options']);
      $output .= '</div>';
    $output .= '</div>';
    $output .= '<div class="standard">';
    $output .= form_render($form_render);
    $output .= '</div>';
  $output .= '</div>';
  return $output;
}
```

bryght

Node form template

```
File : node_form.tpl.php

<div class="node-form">
  <div class="admin">
    <div class="authored">
      <?php print form_render($form['author']) ?>
    </div>
    <div class="options">
      <?php print form_render($form['options']) ?>
    </div>
  </div>
  <div class="standard">
    <?php print form_render($form) ?>
  </div>
</div>
```