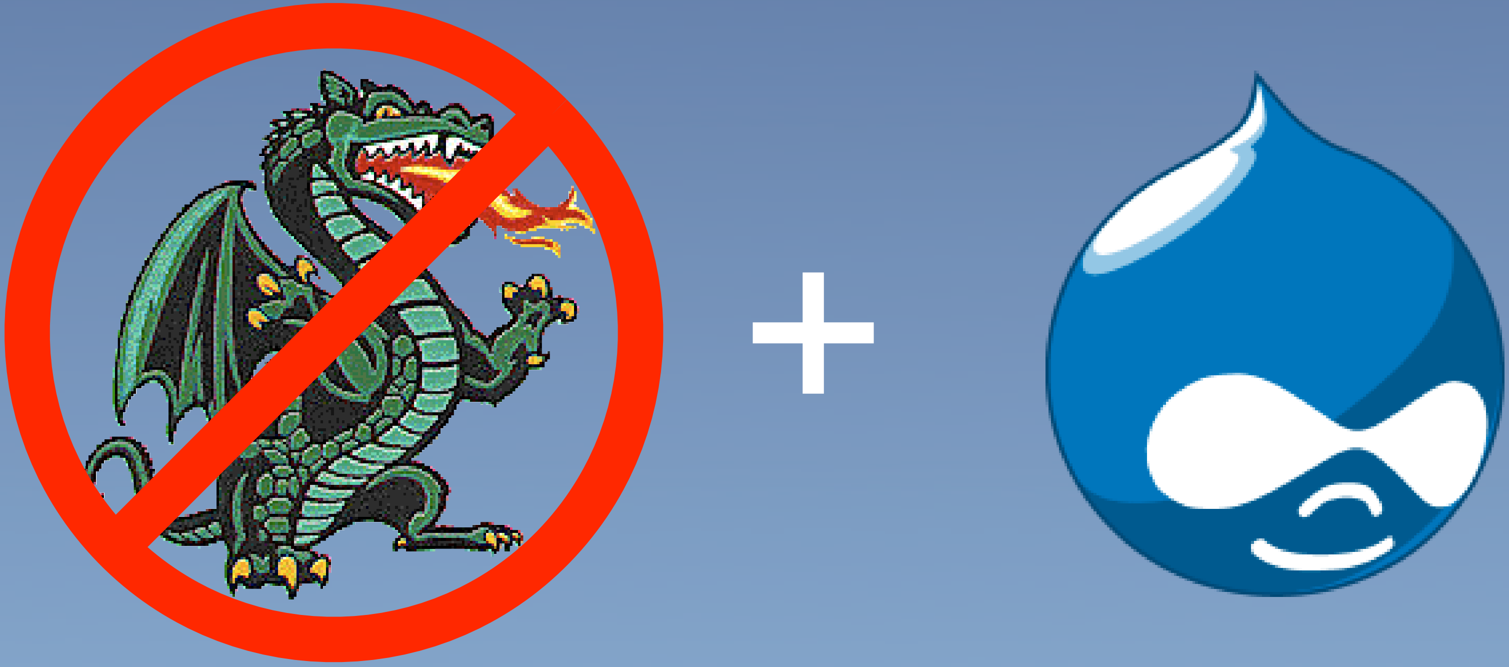
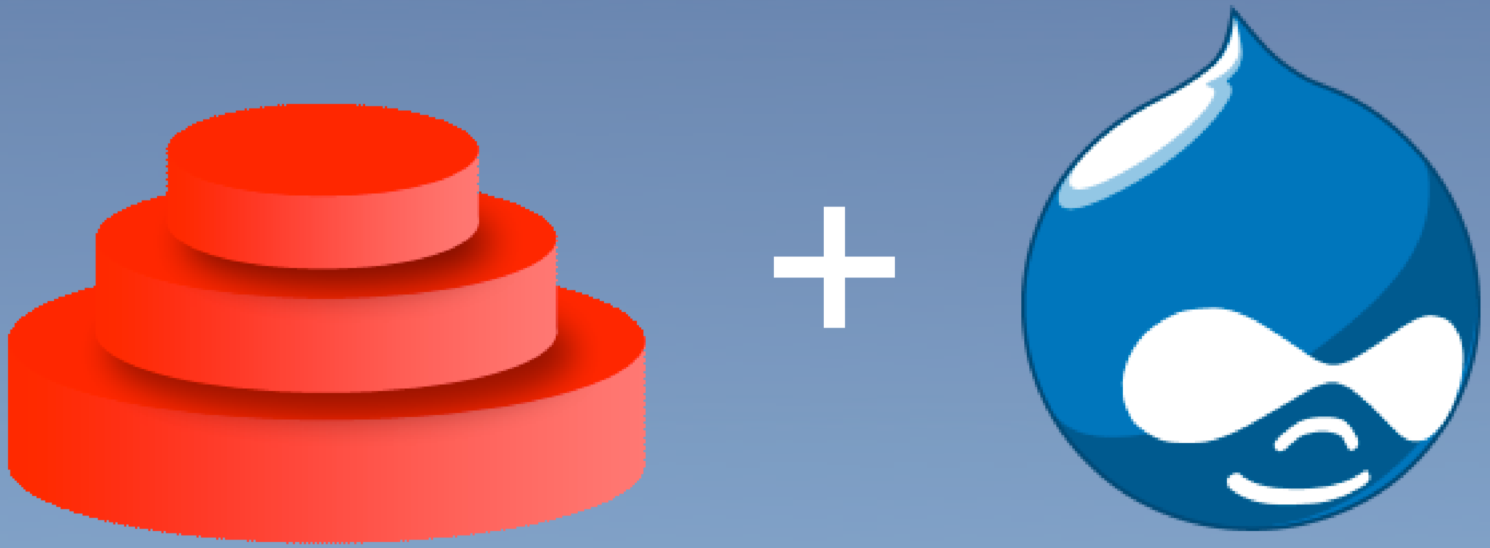
Two New JavaScript Features in Drupal 6

AHAH and Drag and Drop
(JavaScript so you don't have to)

Dragon Drop



Drag and Drop



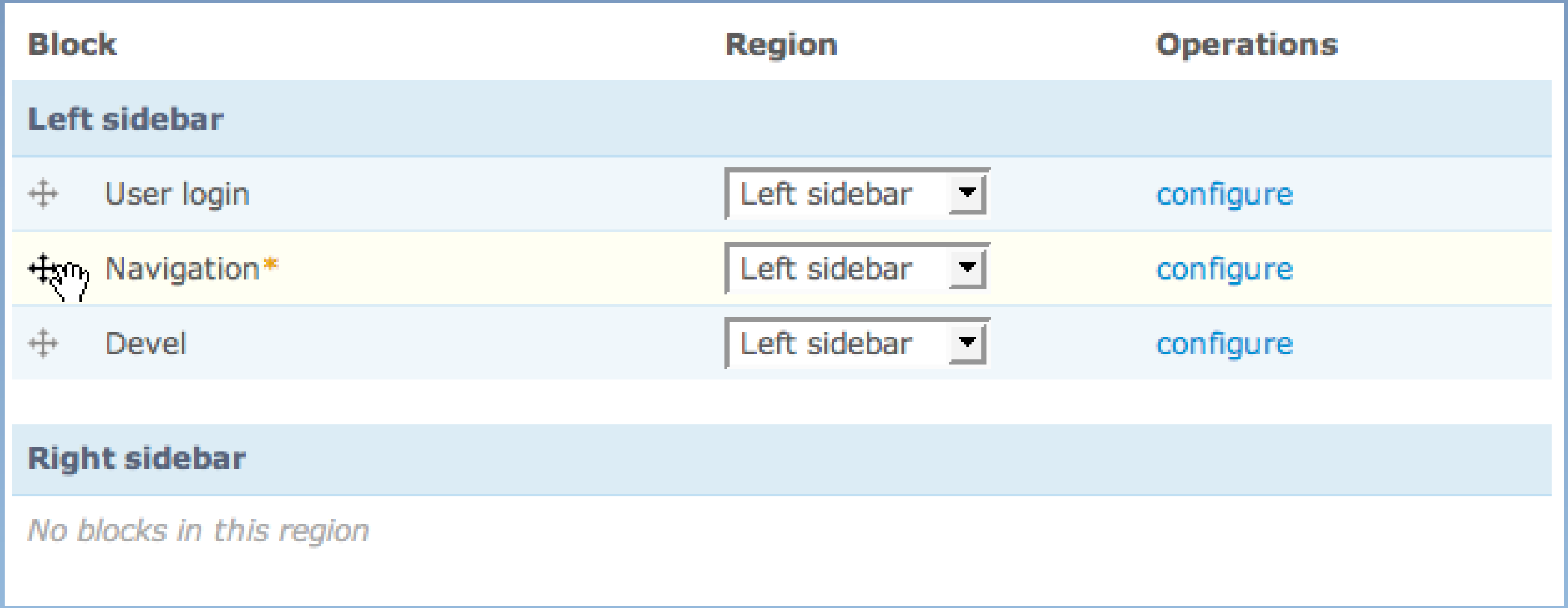
- An easy way of arranging content with JavaScript (jQuery)

Drag and Drop

- Compatible with every major browser:
  - IE (6/7), Firefox 1.5+, Safari 2+, Opera 9, Konqueror
- Allows the sorting of elements **within a table**
- Works to eliminate the word “weight” from Drupal
- Implemented in the theme layer
- Easy to implement in just a few lines of code

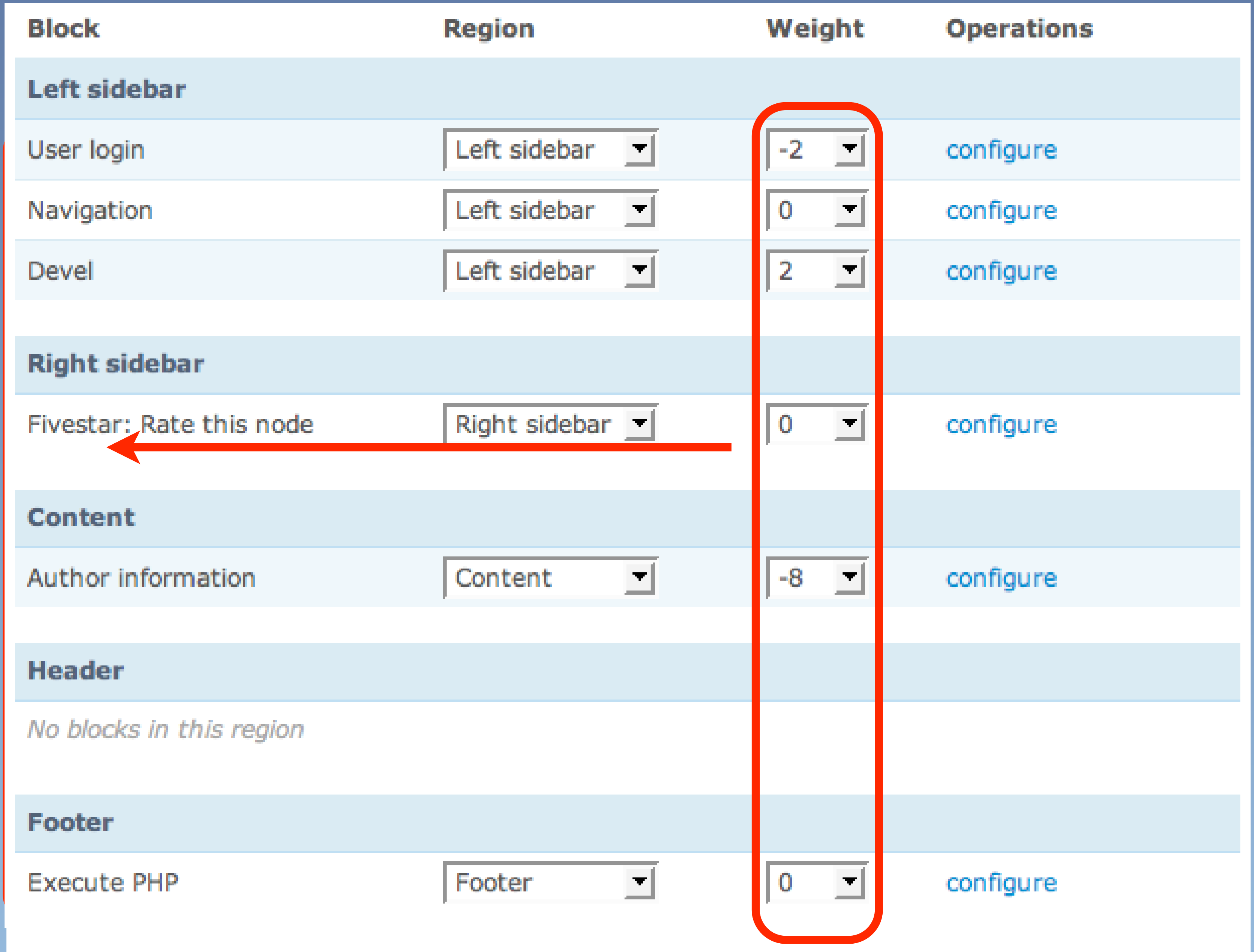
Drag and Drop HTML

- HTML is identical in JavaScript and non-JavaScript versions
- Tabledrag.js makes all changes to the page
- Completely degradable



Block	Region	Operations
Left sidebar		
User login	Left sidebar ▾	configure
Navigation*	Left sidebar ▾	configure
Devel	Left sidebar ▾	configure
Right sidebar		
No blocks in this region		

Drag and Drop HTML



Block	Region	Weight	Operations
Left sidebar			
User login	Left sidebar	-2	configure
Navigation	Left sidebar	0	configure
Devel	Left sidebar	2	configure
Right sidebar			
Fivestar: Rate this node	Right sidebar	0	configure
Content			
Author information	Content	-8	configure
Header			
No blocks in this region			
Footer			
Execute PHP	Footer	0	configure

Kick-ass Demo

Block, Menu

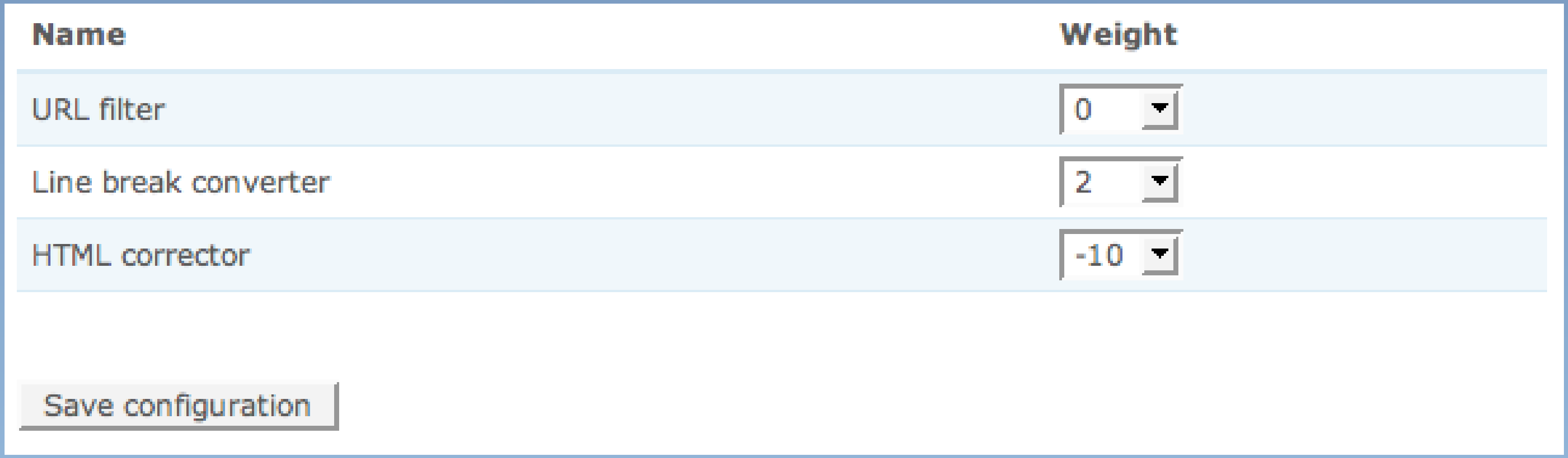
When you want to D&D

1. Create a form for organizing the elements using weight fields.
2. Theme your form into a table.
3. Put weights in an individual column.
4. Add “special” classes.
5. Call drupal_add_tabledrag() from your theme function.

Example: Filter Order

- Filter Module already had weight fields in a table in Drupal 5.

4. Add special classes.
5. Call drupal_add_tabledrag().



Name	Weight
URL filter	0
Line break converter	2
HTML corrector	-10

Special Classes and IDs

- All Drag and Drop tables must have:

  1. An **ID attribute** on the table.
  2. A **draggable** class on draggable rows.
  3. A “group” class on fields in the same column

- In addition, tables with parents can use:

  3. div.**indentation** to set initial indentation.
  4. A **tabledrag-root** class on rows that cannot be indented.
  5. A **tabledrag-leaf** class on rows that cannot have children.

drupal_add_tabledrag()

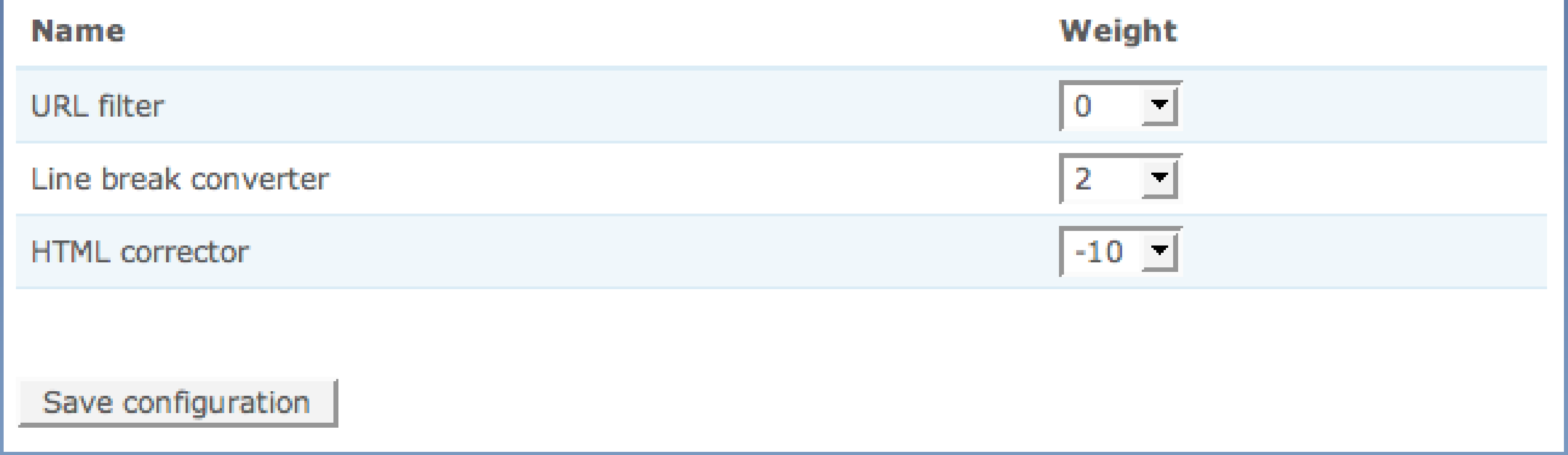
- The “do-it-all” PHP function.

```
drupal_add_tabledrag(
  $table_id,     // The ID of the table.
  $action,       // match or order.
  $relationship, // parent or sibling.
  $group,        // The weight field class.
  $subgroup = NULL, // Used on the blocks page.
  $source = NULL,   // Parent group class.
  $hidden = TRUE,   // Hide the weight column?
  $limit = 0        // Max parent depth.
)
```

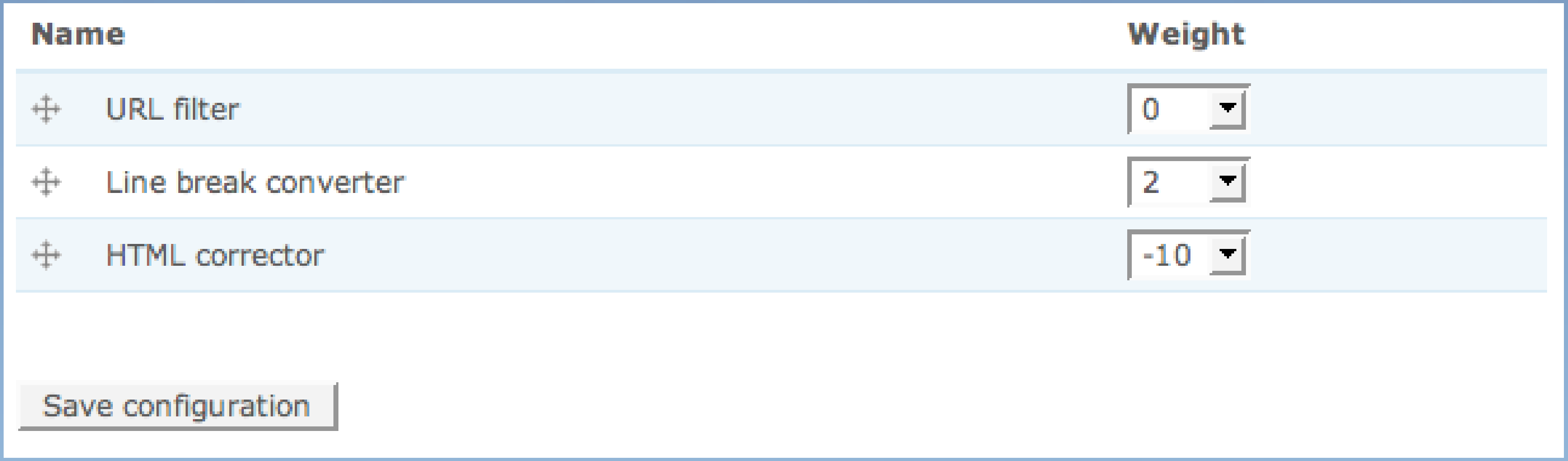
Filter D&D Patch

```
   foreach (element_children($form['names']) as $id) {
     // Don't take form control structures.
     if (is_array($form['names'][$id])) {
-      $rows[] = array(drupal_render($form['names'][$id]), drupal_render($form['weights'][$id]));
+      $form['weights'][$id]['#attributes']['class'] = 'filter-order-weight';
+      $rows[] = array(
+        'data' => array(drupal_render($form['names'][$id]), drupal_render($form['weights'][$id])),
+        'class' => 'draggable',
+      );
     }
   }

-  $output = theme('table', $header, $rows);
+  $output = theme('table', $header, $rows, array('id' => 'filter-order'));
   $output .= drupal_render($form);

+  drupal_add_tabledrag('filter-order', 'order', 'sibling', 'filter-order-weight', NULL, NULL, FALSE);
+
   return $output;
 }
```

Simple Example: Filter Order



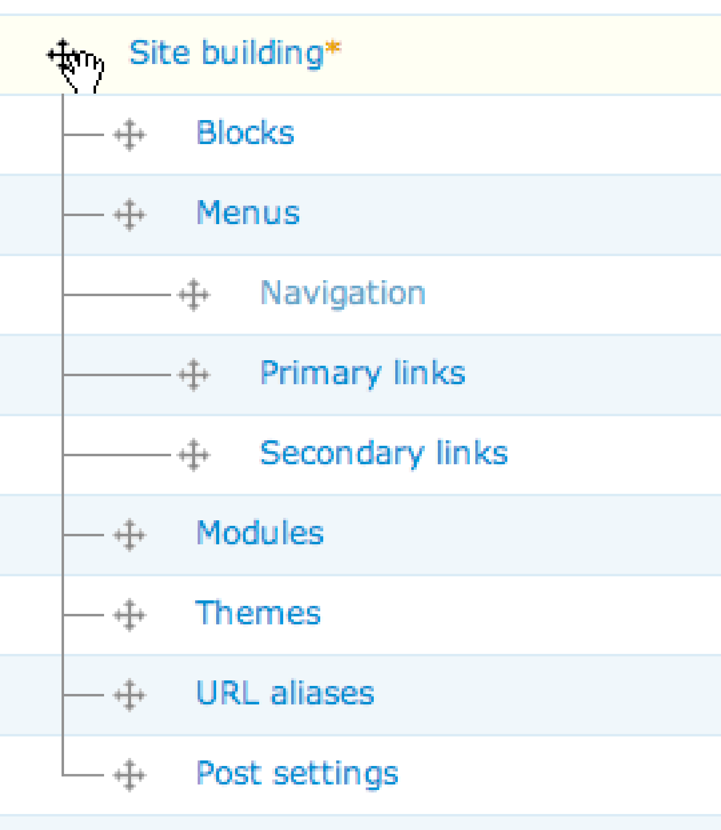
Name	Weight
URL filter	0 ▾
Line break converter	2 ▾
HTML corrector	-10 ▾



Name	Weight
✥ URL filter	0 ▾
✥ Line break converter	2 ▾
✥ HTML corrector	-10 ▾

Tree Structures

- Add two tabledrag instances:
  - Weight ordering
  - Parent relationship



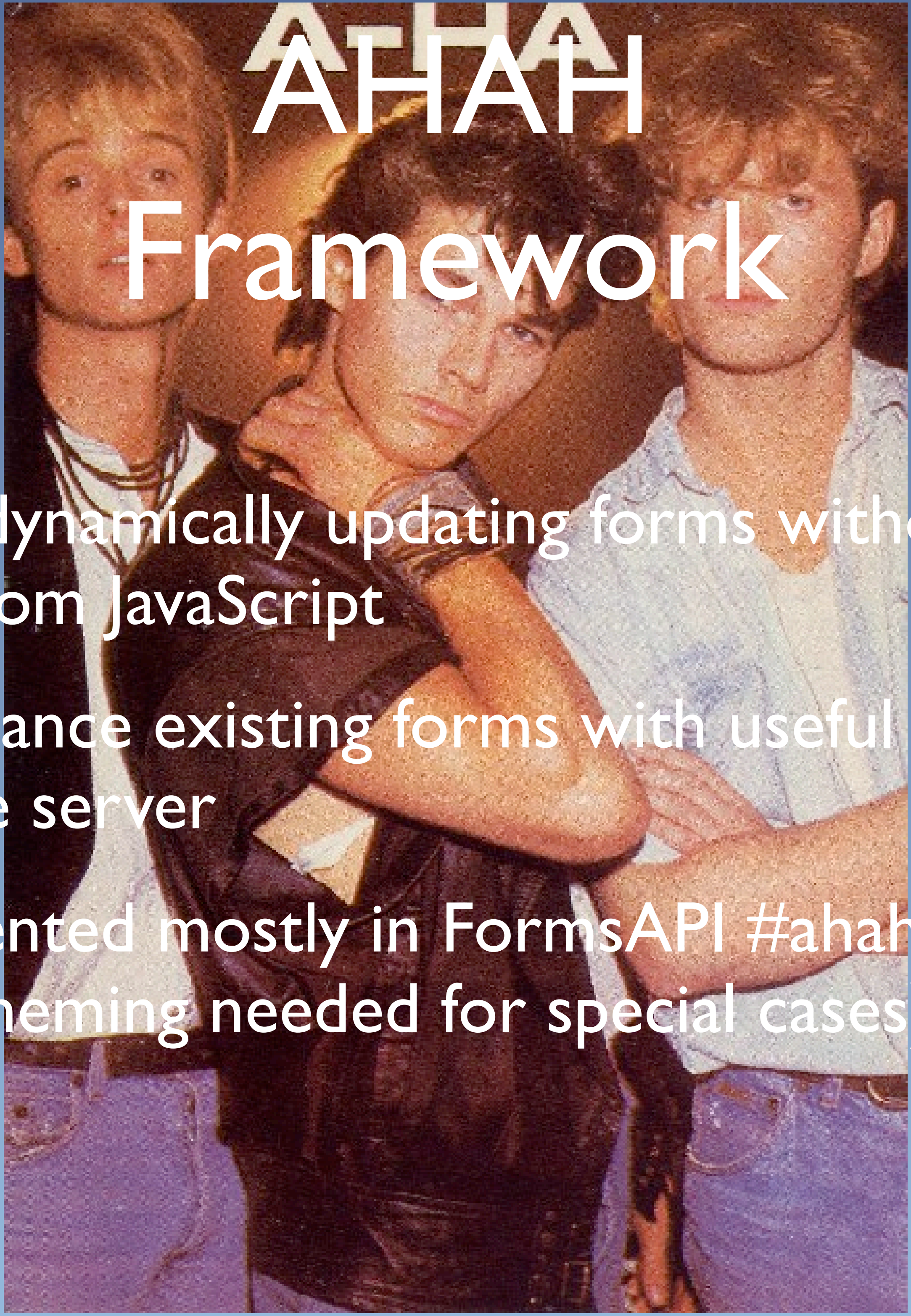
Site building*
- Blocks
- Menus
  - Navigation
  - Primary links
  - Secondary links
- Modules
- Themes
- URL aliases
- Post settings

Tree Structures

```
drupal_add_tabledrag(
  'menu-overview',    // $table_id
  'order',            // $action
  'sibling',          // $relationship
  'menu-weight'       // $group
);
drupal_add_tabledrag(
  'menu-overview',    // $table_id
  'match',            // $action
  'parent',           // $relationship
  'menu-plid',        // $group
  'menu-plid',        // $subgroup
  'menu-mlid',        // $source
  TRUE,               // $hidden
  MENU_MAX_DEPTH - 1  // $limit
);
```

Where can I get it?

- Drupal Core
  - Block Arrangement
  - Menu Structure
  - Taxonomy Terms
  - Upload Attachments
  - Book Outlines
  - Profile Fields
  - Filter Order
- Contrib
  - CCK Fields
  - Webform Fields
  - Views(?)



AHAH Framework

- Allows dynamically updating forms without writing any custom JavaScript
- Can enhance existing forms with useful information from the server
- Implemented mostly in FormsAPI #ahah property (some theming needed for special cases)

Sweet, sweet demo

Upload, CCK

Basic FormsAPI Field

```
$form[‘element’] = array(
  ‘#type’ => ‘textfield’,
  ‘#title’ => t(‘User name’),
);
```

- Forms are structured arrays
- Pound sign (#) means a property of that field

AHAH Enabled Field

```
$form[‘element’] = array(
  ‘#type’ => ‘textfield’,
  ‘#title’ => t(‘User name’),
  ‘#ahah’ => array(
    ‘path’ => ‘mymodule/check_username’,
    ‘wrapper’ => ‘valid_username’,
  ),
);
```

Path and **wrapper** are required #ahah properties.

AHAH Enabled Field

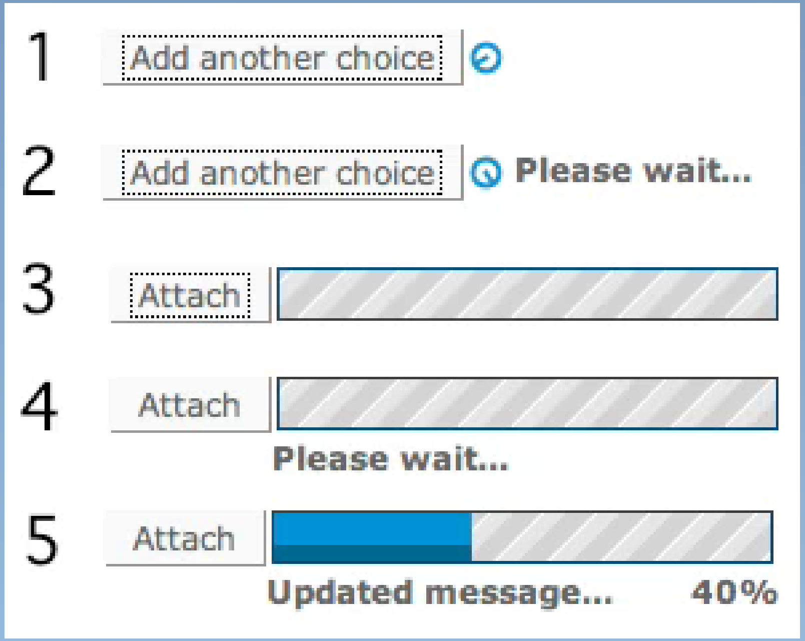
```
$form[‘element’] = array(
  ‘#type’ => ‘textfield’,
  ‘#title’ => t(‘User name’),
  ‘#ahah’ => array(
    ‘path’ => ‘mymodule/check_username’,
    ‘wrapper’ => ‘valid_username’,
  ),
  ‘#suffix’ =>
    ‘<div id=“valid_username”></div>’,
);
```

AHAH Properties

```
‘#ahah’ => array(
  ‘path’ => ‘mymodule/check_username’,
  ‘wrapper’ => ‘valid_username’,
  ‘event’ => ‘change’,
  ‘method’ => ‘replace’,
  ‘progress’ => array(‘type’ => ‘throbber’);
),
```

Extra AHAH options

- #ahah[‘event’] has a reasonable default for form elements so you usually won’t need to set it.
- #ahah[‘progress’] Options



Please wait...

Please wait...

Updated message... 40%

```
array(‘type’ => ‘throbber’)

array(‘type’ => ‘throbber’, ‘message’ => ‘...’);

array(‘type’ => ‘bar’)

array(‘type’ => ‘bar’, ‘message’ => ‘...’);

array(‘type’ => ‘bar’, ‘message’ => ‘...’,
‘path’ => ‘sopme/path’);
```

A Functional Example

```
function regval_form_alter(&$form, $form_state, $form_id) {
  drupal_add_css(drupal_get_path('module', 'regval'). '/regval.css');
  if ($form_id == 'user_register') {
    $form['name']['#ahah'] = array(
      'path' => 'regval/name',
      'wrapper' => 'regval-name-wrapper',
      'progress' => 'none',
      'event' => 'change',
    );
    $form['name']['#suffix'] = '<div id="regval-name-wrapper"></div>';
  }
}
```

A Functional Example

```
function regval_menu() {
  $items['regval/name'] = array(
    'page callback' => 'regval_check_name',
    'access callback' => 'regval_access_callback',
    'type' => MENU_CALLBACK,
  );
}
```

A Functional Example

```
function regval_check_name() {
  $name = $_POST['name'];
  if ($name != '' && !is_null($name)) {
    $exists = db_result(db_query("SELECT uid FROM {users} WHERE
                                  name = '%s' AND uid != %d", $name, $user->uid));
    if ($exists) {
      $output = theme('image', 'misc/watchdog-error.png', t('error')) .' '.
                t('Username already exists.');
    }
    else {
      $output = theme('image', 'misc/watchdog-ok.png', t('ok')) .' '.
                t("Username is available.");
    }
    print drupal_json(array('result' => TRUE, 'data' => $output));
  }
  else {
    print '';
  }
}
```

Wee! demo

Regval

Updating a Form

- Such as:
  - A “more” button to add fields
  - Drill-down structure
  - Select list form changes
- Take into account FormsAPI security

Updating a Form

1. Retrieve form cache from database
2. Add additional fields to the database version of the form
3. Save the new form back to the database
4. Render the new part of the form and return it back to the original page

Updating a Form

```
function poll_choice_js() {
  $delta = count($_POST['choice']);

  ...

  $form = form_get_cache($form_build_id, $form_state);
  $form['choice_wrapper']['choice'][$delta] = _poll_choice_form($delta);
  form_set_cache($form_build_id, $form, $form_state);

  ...

  drupal_json(array('status' => TRUE, 'data' => $output));
}
```

- Form is updated on AHAH requests to maintain form security

Goals for the Future

- Remove ‘path’ #ahah property and replace with ‘callback’
- Easily render small pieces of a form for AHAH form updates

Questions

D&D and AHAH
Developers and Testers

- fajerstarter
- Stefan Nagtegaal
- scor
- catch
- nedjo
- eaton
- pwolanin
- chx
- Wim Leers
- webchick
- bdragon
- beginner
- Gábor Hojtsy
- Dries

Many others. Thanks!