

#type	#access	#ahah	#action	#autocomplete_path	#button_type	#collapsed	#collapsible	#cols	#default_value	#delta	#description	#disabled	#element_validate	#execute_submit_callback	#field_prefix	#field_suffix	#maxlength	#method	#multiple	#options	#parents	#redirect	#required	#return_value	#rows	#size	#src	#submit	#title	#tree	#validate	#value	#weight
checkbox	X	X						X	X	X	X	X								X		X	X					X	X			X	
checkboxes	X							X	X	X	X	X							X	X		X	X					X	X			X	
date	X							X	X	X	X										X	X						X	X			X	
fieldset	X				X	X					X		X								X							X	X			X	
file	X										X	X	X								X							X	X			X	
password	X	X									X	X	X								X		X					X	X			X	
radio	X	X						X	X	X	X										X	X	X					X	X			X	
radios	X							X	X	X	X								X	X		X	X					X	X			X	
select	X	X						X	X	X	X							X	X	X		X			X			X	X			X	
textarea	X	X					X	X	X	X	X										X		X	X				X	X			X	
textfield	X	X	X					X	X	X	X		X	X	X						X		X		X			X	X			X	
button		X		X									X	X							X							X	X	X	X	X	
imagebutton		X		X									X	X							X					X	X	X	X	X	X	X	
submit		X		X									X	X							X							X	X	X	X	X	
form			X														X				X						X	X	X				
hidden								X					X																X				
markup													X																		X	X	
item											X		X															X			X	X	
value													X								X							X		X			
weight	X							X	X	X	X	X									X	X						X	X			X	

Allowed for any type but 'value': #after_build, #attributes, #prefix, #suffix, #theme
 Source: http://api.drupal.org/api/file/developer/topics/forms_api_reference.html/6

FORM FUNCTIONS

drupal_get_form(\$form_id)

Retrieve and process a form by its ID.

form_set_error(\$element, \$msg)

Set an error message on a particular element in a form.

FORM HOOKS

hook_form_alter(&\$form, \$form_state, \$form_id)

Alter any form. The \$form_id variable indicates which form is being processed.

hook_form_[FORM_ID]_alter(&\$form, \$form_state)

Alter a specific form with a form ID of [FORM_ID]. For example, if the form is named 'foo', the above hook would be hook_form_foo_alter(&\$form, \$form_state).

hook_forms()

Define multiple forms at once using an associative array of form IDs and form definitions.

FORM EXAMPLE

```
// Create the form.
<?php
function my_form(&$form_state) {
  $form['my_field'] = array(
    '#type' => 'textfield',
    '#title' => t("Enter a Number (1-10)"),
    '#size' => 4,
  );

  $form['my_submit'] = array(
    '#type' => 'submit',
    '#value' => t("Click Here to Submit"),
  );
  return $form;
}
```

```
// Validate form input.
function my_form_validate($form, $form_state) {
  // Get the value.
  $my_field = (int)$form_state['values']['my_textfield'];

  // Check the value and return an error if it is not in range.
  if ($my_field < 1 || $my_field > 10) {
    form_set_error('my_field', 'Enter a val between 1 and 10');
  }
}

// Submit the form.
function my_form_submit($form, &$form_state) {
  // Store value.
  variable_set('my_field', $form_state['values']['my_field']);
  // Notify user.
  drupal_set_message('The value has been saved.', 'status');
}
?>
```



HOOKS

function hook_menu()

Purpose: Define router items for handling requests.

```
<?php
function hook_menu() {
  $items['mypath/%object'] = array(
    'title' => 'Page title',
    'title arguments' => array(),
    'title callback' => 't',
    'description' => 'Your description goes here.',
    'access callback' => 'object_check_access',
    'access arguments' => array(),
    'page arguments' => array(),
    'page callback' => 'object_display',
    'block callback' => '',
    'menu_name' => NULL,
    'tab_parent' => NULL,
    'tab_root' => NULL,
    'file' => 'name_of_file.inc',
    'file path' => drupal_get_path('module', 'name_of_module_goes_here'),
    'weight' => 0,
    'type' => MENU_NORMAL_ITEM,
  );
  return $items;
}
?>
```

Further reading: <http://drupal.org/node/102338>

function hook_menu_alter(&\$callbacks)

Purpose: Alter a menu router item defined by another module's hook_menu().

```
<?php
function hook_menu_alter(&$callbacks) {
  // Example - disable the page at node/add
  $callbacks['node/add']['access callback'] = FALSE;
}
?>
```

function hook_theme(\$existing, \$type, \$theme, \$path)

Purpose: Define a themeable element.

For a theme function:

```
<?php
function hook_theme($existing, $type, $theme, $path) {
  $items['status_report'] = array(
    'arguments' => array('requirements' => NULL),
    'file' => 'system.admin.inc',
  );
  return $items;
}
?>
```

For a template:

```
<?php
function hook_theme($existing, $type, $theme, $path) {
  $items['node'] = array(
    'arguments' => array('node' => NULL, 'teaser' => FALSE, 'page' => FALSE),
    'template' => 'node',
  );
  return $items;
}
?>
```

Further reading: <http://drupal.org/theme-guide/6>

function hook_theme_registry_alter(&\$theme_registry)

Purpose: Alter a themeable element's definition.

```
<?php
function hook_theme_registry_alter(&$theme_registry) {
  // Kill the next/previous forum topic navigation links.
  foreach ($theme_registry['forum_topic_navigation']['preprocess
functions'] as $key => $value) {
    if ($value == 'template_preprocess_forum_topic_navigation') {
      unset($theme_registry['forum_topic_navigation']['preprocess
functions'][$key]);
    }
  }
}
?>
```

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

?

```
function hook_nodeapi(&$node, $op, $a3 = NULL,
$a4 = NULL)
```

Purpose: Operate on or respond to a step in the node lifecycle.

```
<?php
function hook_nodeapi(&$node, $op, $a3 = NULL, $a4 = NULL) {
  switch ($op) {
    case 'load':
      // The node is being loaded. Return extra data to merge into $node.
      case 'view':
      // The node is about to be rendered. Alter the $node->content array.
      $a3 and $a4 are $teaser and $page.
      case 'alter':
      // The node has been rendered. Alter $node->body or $node->teaser.
      case 'prepare':
      // The node is about to be shown on add/edit forms.
      case 'validate':
      // Validate a node form. $a3 is $form.
      case 'presave':
      // Make post-validation changes to a node that don't get revalidated.
      case 'insert':
      // A new node (revision) is being saved. Save your own data.
      case 'update':
      // An existing node (revision) is being saved. Save your own data.
      case 'delete':
      // The entire node is being deleted.
      case 'delete revision':
      // Just one revision of a node is being deleted.
      case 'print':
      // Prepare the node for printing.
      case 'update index':
      // Return extra info for the search index.
      case 'search result':
      // Return extra display information for search result pages.
      case 'prepare translation':
      // Modify the node for translation.
      case 'rss item':
      // Return additional properties for this node in an RSS feed.
    }
  }
}
?>
```

JAVASCRIPT

Add an existing JS file to the page.

```
<?php
drupal_add_js(
  drupal_get_path('module', 'example') . '/example.js', // Path to the script
  $type, // either "module" or "theme". Default "module".
  $scope, // either "header" or "footer". Default "header".
  $defer, // TRUE to set file to deferred in IE. Default FALSE.
  $cache, // TRUE to allow browser to cache file. Default TRUE.
  $preprocess, // TRUE to aggregate into one JS file. Default TRUE.
);
?>
```

Pass settings data to Javascript

```
$settings['example']['mysetting'] = 'value';
drupal_add_js($settings, 'setting');
In JS, access at: Drupal.settings.example.mysetting;
```

URLS

Get just a URL or path

```
<?php
$url = url($path, array(
  'query' => array of GET query values,
  'fragment' => 'fragment/anchor name without #',
  'absolute' => TRUE to force full URL. Default FALSE,
  'alias' => TRUE to skip alias lookup. Default FALSE,
  'external' => TRUE if $path is a non-Drupal URL. Default FALSE,
  'language' => language object if relevant,
));
?>
```

Get an <a> link

```
<?php
$link = l($link_text, $path, array(
  'attributes' => array of HTML attributes,
  'query' => array of GET values,
  'fragment' => 'fragment/anchor name without #',
  'absolute' => TRUE to force full URL. Default FALSE,
  'alias' => TRUE to skip alias lookup. Default FALSE,
  'html' => TRUE to allow HTML in $link_text. Default FALSE,
));
?>
```

SECURITY FUNCTIONS

check_plain(\$text)

Make \$text into plain text, escaping HTML to show to the user.

check_markup(\$text, \$format = FILTER_FORMAT_DEFAULT, \$check = TRUE)

Run \$text through a defined input format filter set. \$format is an integer. \$check refers to access controls.

filter_xss(\$string, \$allowed_tags = array())

Strip out all HTML tags but those in \$allowed_tags. A smarter version of strip_tags().

filter_xss_admin(\$string)

filter_xss() with a very permissive set of allowed tags.

STRING FUNCTIONS

Use these instead of the normal PHP versions to support non-Western characters.

drupal_strlen(\$text)

Returns the number of characters in \$text.

drupal_strtolower(\$text)

Returns the lowercase version of a string.

drupal_strtoupper(\$text)

Returns the uppercase version of a string.

drupal_substr(\$text, \$start, \$length = NULL)

UTF-8-safe equivalent of substr()

drupal_ucfirst(\$text)

Capitalize the first letter of a UTF-8 string.

truncate_utf8(\$string, \$len, \$wordsafe = FALSE, \$dots = FALSE)

Truncate \$string to \$len characters, or the previous word boundary if \$wordsafe, appending an ellipsis if \$dots.

SECURE QUERIES

Never ever inject user-supplied data into a query directly. Always use a placeholder:

```
%s: A string
%d: Integer
%f: Floating point number
%b: Binary data
%%: A literal % value
```

Always enclose tables in {} : {tablename}

Normal query

```
$result = db_query("SELECT foo FROM {table} WHERE int_field=%d AND float_field=%f AND string_field=%s", array(5, 2.5, 'hello'));
Select only the first 5 results.
```

```
$result = db_query_range("SELECT foo FROM {table} WHERE int_field=%d AND float_field=%f AND string_field=%s", array(5, 2.5, 'hello'), 0, 5);
```

Fetching results

```
<?php
$first_element = db_result($result);
$record_as_object = db_fetch_object($result);
$record_as_array = db_fetch_array($result);
while ($record = db_fetch_object($result)) {
  // ...
}
?>
```

STRINGS IN CODE

```
<?php
t('Some @escaped text, and %placeholder text, and !unescaped text',
array(
  '@escaped' => 'Run through check_plain()',
  '%placeholder' => 'Run through theme_placeholder()',
  '!unescaped' => 'Inserted literally. Be careful.',
));
?>
```