```
add_settings_field( string $id, string $title, callable $callback,
string $page, string $section = 'default', array $args = array() )
```

Adds a new field to a section of a settings page.

Description

Part of the Settings API. Use this to define a settings field that will show as part of a settings section inside a settings page. The fields are shown using do_settings_fields() in do_settings_sections() .

The $callback argument should be the name of a function that echoes out the HTML input tags for this setting field. Use get_option() to retrieve existing values to show.

Parameters

$id string required

Slug-name to identify the field. Used in the `'id'` attribute of tags.

$title string required

Formatted title of the field. Shown as the label for the field during output.

$callback callable required

Function that fills the field with the desired form inputs. The function should echo its output.

$page string required

The slug-name of the settings page on which to show the section (general, reading, writing, ...).

$section string optional

The slug-name of the section of the settings page in which to show the box. Default `'default'` .

Default: `'default'`

$args array optional

Extra arguments that get passed to the callback function.

> **label_for** `string`
>
> When supplied, the setting title will be wrapped in a `<label>` element, its `for` attribute populated with this value.
>
> **class** `string`
>
> CSS Class to be added to the `<tr>` element when the field is output.

Default: `array()`

More Information

You MUST register any options used by this function with register_setting() or they won't be saved and updated automatically.

The callback function needs to output the appropriate html input and fill it with the old value, the saving will be done behind the scenes.

The html input field's name attribute must match $option_name in register_setting(), and value can be filled using get_option().

This function can also be used to add extra settings fields to the default WP settings pages like media or general. You can add them to an existing section, or use add_settings_section() to create a new section to add the fields to.

See Settings API for details.

Source

wp-admin/includes/template.php

Expand code Copy

```
function add_settings_field( $id, $title, $callback, $page, $section = 'default', $args = array()
    global $wp_settings_fields;

    if ( 'misc' === $page ) {
        _deprecated_argument(
            __FUNCTION__,
            '3.0.0',
            sprintf(
                /* translators: %s: misc */
                __( 'The "%s" options group has been removed. Use another settings group.' ),
                'misc'
            )
        );
```

View all references·View on Trac·View on GitHub

Related

Uses

__()
wp-includes/l10n.php

_deprecated_argument()
wp-includes/functions.php

Changelog

Version	Description
4.2.0	The `$class` argument was added.
2.7.0	Introduced.

User Contributed Notes

Codex9 years ago 6

With Label

Adds a setting with id `myprefix_setting-id` to the General Settings page. `myprefix` should be a unique string for your plugin or theme. Sets a label so that the setting title can be clicked on to focus on the field.

Copy

```
add_settings_field( 'myprefix_setting-id',
    'This is the setting title',
    'myprefix_setting_callback_function',
    'general',
    'myprefix_settings-section-name',
    array( 'label_for' => 'myprefix_setting-id' ) );
```

Log in to add feedback

tradesouthwest4 years ago 2

A checkbox settings field can be checked on the front end by simply looking for isset. No need to add additional checks like 1, 0, true, false.... if a checkbox is not set then it returns false.

Expand code Copy

```
/* ***************** CHECKBOXES ***************** */
    // settings checkbox
    add_settings_field(
        'wpdevref_removestyles_field',
        esc_attr__('Remove Plugin Styles', 'wpdevref'),
        'wpdevref_removestyles_field_cb',
        'wpdevref_options',
        'wpdevref_options_section',
        array(
            'type'         => 'checkbox',
            'option_group' => 'wpdevref_options',
            'name'         => 'wpdevref_removestyles_field',
            'label_for'    => 'wpdevref_removestyles_field',
```

Then the callback would be added as such:

Expand code Copy

```
/**
 * switch for 'remove styles' field
 * @since 2.0.1
 * @input type checkbox
 */
function wpdevref_removestyles_field_cb($args)
{
    $checked = '';
    $options = get_option($args['option_group']);
    $value   = ( isset( $options[$args['name']] ) )
```

```
    $value    = ( !isset( $options[$args['name']] ) )
                ? null : $options[$args['name']];
    if($value) { $checked = ' checked="checked" '; }
        // Could use ob_start.
```

And checking to render action on the front side would use:

Copy

```
// Options getter could be a function with arguments.
    $options = get_option('wpdevref_options');
    $value   = ( !isset($options['wpdevref_removestyles_field'] ) )
                ? '' : $options['wpdevref_removestyles_field'];
    if ( !$value ) {
    // Do some magic
    }
```

Optionally you can add a 'false' into any conditional (empty, null, '', 0).

Log in to add feedback

princepink 9 years ago 1

I suspect Used in the 'id' attribute of tags might be rewritten to Used in the 'name' attribute of tags.
I think the $id param is used for identifying the field to be recognised by WP and to show the field or get that's value. Whether to be used as actual tag's attribute id's value or not depends on the circumstances (in $callback). Normally the name attribute might be taken for this aim.
I just had been confused this param means to generate the attribute for some form element, but it seems not. When put 'label_for' in the $args that will be passed to the $callback, this generates label tag with attribute for automatically. So this value should be same as an actual id's value in the $callback which you write.

Log in to add feedback

tehlivi 7 years ago 0

Expand code Copy

```
add_action('admin_init', 'your_function');
function your_function(){
    add_settings_field(
        'myprefix_setting-id',
        'This is the setting title',
        'myprefix_setting_callback_function',
        'general',
        'default',
        array( 'label_for' => 'myprefix_setting-id' )
    );
}

function myprefix_setting_callback_function($args){
```

Log in to add feedback

tehlivi 7 years ago 0

Object Oriented:

Expand code Copy

```
class ClassName {
    public function __construct() {
        add_action( 'admin_init', array( $this, 'your_function' ) );
    }

    function your_function() {
        add_settings_field(
            'myprefix_setting-id',
            'This is the setting title',
            array( $this, 'myprefix_setting_callback_function' ),
            'general',
            'default',
            array( 'label_for' => 'myprefix_setting-id' ),
```

Show feedback (1)Log in to add feedback

Bence Szalai 5 years ago 0

The `$id` argument description says “Used in the ‘id’ attribute of tags”, however this means you have to ensure this **`$id`** is used as the HTML **`id`** tag of your **`input`** element related to the field. WP only use this `$id` to have an unique key for your field in it’s internal settings_field list (`$wp_settings_fields`).
As WP does not control the way the input element is added to your Admin HTML, you have to ensure you output the input element with an `id` that matches the `$id` tag. This can be done by configuring the `$callback` to a function, that will produce the correct `input` element with the correct `id` tag.
The ‘label_for’ element in the $args array should also match the very same `id` in order for the browser to understand which `label` belongs to which `input` field.

It worth noting also, that the `id` tag of the `input` element should also match the $option_name (2nd) parameter you are using in your `register_setting()` call, otherwise the Settings API will fail to match the value sent by the browser in `$_POST` to your setting, and your setting will never be saved.

So long story short, we have a bunch of different names and arguments, but basically `$id` and `$args['label_for']` of `add_settings_field()` call and the `$option_name` of `register_setting()` call PLUS the `id` you use in your input field callback, should all be the same, unique **`id`** . Also the same `id` should be used as the `$option` parameter in the `get_option($option)` calls to get the value of the setting.

Expand code Copy

```
register_setting( 'mygroup', 'mynewcheckboxID' );
add_settings_field(
        'mynewcheckboxID',
        'My New Checkbox',
```

```
        'callback_input_myid',
        'myAdminPage',
        'myAdminSection',
        [
            'label_for' => 'mynewcheckboxID'
        ] );

function callback_input_myid() {
    echo "<input type='checkbox' id='mynewcheckboxID' value='1'"
```

Show feedback (1)Log in to add feedback

You must log in before being able to contribute a note or feedback.