This repository Search Pull requests Issues Gist

WebDevStudios / CMB2 Watch 115 Star 918 Fork 231

Code Issues 151 Pull requests 24 Wiki Pulse Graphs

Field Types

Edit New Page

jmarceli edited this page 5 days ago · 97 revisions

Here's the built-in fields you can include in your metabox. You can also add your own field types.

Note that all the id fields should have proper prefixes. It's a good practice to create a unique prefix for your fields so you don't risk using the same id as another theme/plugin. Take a look at example-functions.php to see how you can define the prefix.

Not all built-in fields have been 100% documented, so please see the example file for additional details.

Types:

1. `title` An arbitrary title field *
2. `text`
3. `text_small`
4. `text_medium`
5. `text_email`
6. `text_url`
7. `text_money`
8. `textarea`
9. `textarea_small`
10. `textarea_code`
11. `text_time` Time picker
12. `select_timezone` Time zone dropdown
13. `text_date_timestamp` Date Picker (UNIX timestamp)
14. `text_datetime_timestamp` Text Date/Time Picker Combo (UNIX timestamp)
15. `text_datetime_timestamp_timezone` Text Date/Time Picker/Time zone Combo (serialized DateTime object)
16. `hidden` Hidden input type
17. `colorpicker` Color picker
18. `radio` *
19. `radio_inline` *
20. `taxonomy_radio` *
21. `taxonomy_radio_inline` *
22. `select`
23. `taxonomy_select` *
24. `checkbox` *
25. `multicheck` and `multicheck_inline`
26. `taxonomy_multicheck` *
27. `taxonomy_multicheck_inline`
28. `wysiwyg` (TinyMCE) *
29. `file` Image/File upload *†

Pages 14

Home
Adding metaboxes to user profile
Adding your own field types
Adding your own show_on filters
Basic Usage
Bringing Metaboxes to the Front end
Create New Posts (or Custom Post Type) Entries Using A Front End Form
Display Options
Field Types
Notable Changes in CMB2
Plugin code to add JS validation of "required" fields
Tips & Tricks
Troubleshooting
Using CMB to create an Admin Theme Options Page

Clone this wiki locally

https://github.com/WebDe

Clone in Desktop

30. `file_list` Image/File list upload
31. `oembed` Converts oembed urls (instagram, twitter, youtube, etc. oEmbed in the Codex)
32. `group` Hybrid field that supports adding other fields as a repeatable group. *

More Info

- Create your own field type
- Common field parameters shared by all fields

* Not available as a repeatable field

† Use `file_list` for repeatable

`title`

A large title (useful for breaking up sections of fields in metabox). Example:

```
$cmb->add_field( array(
    'name' => 'Test Title',
    'desc' => 'This is a title description',
    'type' => 'title',
    'id'   => 'wiki_test_title'
) );
```

CSS Field Class:

```
cmb-type-title
```

`text`

Standard text field (large). Example:

```
$cmb->add_field( array(
    'name'    => 'Test Text',
    'desc'    => 'field description (optional)',
    'default' => 'standard value (optional)',
    'id'      => 'wiki_test_text',
    'type'    => 'text',
) );
```

CSS Field Class:

```
cmb-type-text
```

`text_small`

Small text field. Example:

```
$cmb->add_field( array(
    'name'    => 'Test Text Small',
    'desc'    => 'field description (optional)',
    'default' => 'standard value (optional)',
    'id'      => 'wiki_test_textsmall',
```

```
    'type'    => 'text_small'
) );
```

CSS Field Class:

```
cmb-type-text-small
```

text_medium

Medium text field. Example:

```
$cmb->add_field( array(
    'name'    => 'Test Text Medium',
    'desc'    => 'field description (optional)',
    'default' => 'standard value (optional)',
    'id'      => 'wiki_test_textmedium',
    'type'    => 'text_medium'
) );
```

CSS Field Class:

```
cmb-type-text-medium
```

text_email

Standard text field which enforces an email address. Example:

```
$cmb->add_field( array(
    'name' => 'Test Text Email',
    'id'   => 'wiki_test_email',
    'type' => 'text_email',
) );
```

CSS Field Class:

```
cmb-type-text-email
```

text_url

Standard text field which enforces a url. Example:

```
$cmb->add_field( array(
    'name' => __( 'Website URL', 'cmb' ),
    'id'   => 'wiki_test_facebookurl',
    'type' => 'text_url',
    // 'protocols' => array( 'http', 'https', 'ftp', 'ftps', 'mailto', 'news', 'irc'
) );
```

CSS Field Class:

```
cmb-type-text-url
```

text_money

Standard text field with dollar sign in front of it (useful to prevent users from adding a dollar sign to input). Example:

```
$cmb->add_field( array(
    'name' => 'Test Money',
    'desc' => 'field description (optional)',
    'id' => 'wiki_test_textmoney',
    'type' => 'text_money',
    // 'before_field' => '£', // Replaces default '$'
) );
```

CSS Field Class:

```
cmb-type-text-money
```

textarea

Standard textarea. Example:

```
$cmb->add_field( array(
    'name' => 'Test Text Area',
    'desc' => 'field description (optional)',
    'default' => 'standard value (optional)',
    'id' => 'wiki_test_textarea',
    'type' => 'textarea'
) );
```

CSS Field Class:

```
cmb-type-textarea
```

textarea_small

Smaller textarea. Example:

```
$cmb->add_field( array(
    'name' => 'Test Text Area Small',
    'desc' => 'field description (optional)',
    'default' => 'standard value (optional)',
    'id' => 'wiki_test_textareasmall',
    'type' => 'textarea_small'
) );
```

CSS Field Class:

```
cmb-type-textarea-small
```

textarea_code

Code textarea. Example:

```
$cmb->add_field( array(
    'name' => 'Test Text Area Code',
    'desc' => 'field description (optional)',
    'default' => 'standard value (optional)',
    'id' => 'wiki_test_textareacode',
    'type' => 'textarea_code'
) );
```

CSS Field Class:

```
cmb-type-textarea-code
```

text_time

Time picker field. Example:

```
$cmb->add_field( array(
    'name' => 'Test Date Picker',
    'id' => 'wiki_test_texttime',
    'type' => 'text_time'
    // 'time_format' => 'h:i:s A',
) );
```

CSS Field Class:

```
cmb-type-text-time
```

Custom Field Attributes:

- `time_format` , defaults to 'h:i A'. See php.net/manual/en/function.date.php.

select_timezone

Timezone field. Example:

```
$cmb->add_field( array(
    'name' => 'Time zone',
    'id'   => 'wiki_test_timezone',
    'type' => 'select_timezone',
) );
```

CSS Field Class:

```
cmb-type-select-timezone
```

text_date_timestamp

Date field, stored as UNIX timestamp. Useful if you plan to query based on it (ex: events listing).
Example:

```
$cmb->add_field( array(
    'name' => 'Test Date Picker (UNIX timestamp)',
    'id'   => 'wiki_test_textdate_timestamp',
    'type' => 'text_date_timestamp',
    // 'timezone_meta_key' => 'wiki_test_timezone',
    // 'date_format' => 'l jS \of F Y',
) );
```

CSS Field Class:

```
cmb-type-text-date-timestamp
```

Alias:

```
text_date
```

Custom Field Attributes:

- `timezone_meta_key` , Optionally make this field honor the timezone selected in the `select_timezone` field specified above.
- `date_format` , defaults to 'm/d/Y'. See php.net/manual/en/function.date.php.

text_datetime_timestamp

Date and time field, stored as UNIX timestamp. Example:

```
$cmb->add_field( array(
    'name' => 'Test Date/Time Picker Combo (UNIX timestamp)',
    'id'   => 'wiki_test_datetime_timestamp',
    'type' => 'text_datetime_timestamp',
) );
```

CSS Field Class:

```
cmb-type-text-datetime-timestamp
```

text_datetime_timestamp_timezone

Date, time and timezone field, stored as serialized DateTime object. Example:

```
$cmb->add_field( array(
    'name' => 'Test Date/Time Picker/Time zone Combo (serialized DateTime object)',
    'id'   => 'wiki_test_datetime_timestamp_timezone',
    'type' => 'text_datetime_timestamp_timezone',
) );
```

CSS Field Class:

```
cmb-type-datetime-timestamp-timezone
```

hidden

Adds a `hidden` input type to the bottom of the CMB2 output. Example:

```
$cmb->add_field( array(
    'id'   => 'wiki_test_hidden',
    'type' => 'hidden',
) );
```

CSS Field Class:

not applicable to this field type.

colorpicker

A colorpicker field. Example:

```
$cmb->add_field( array(
    'name'    => 'Test Color Picker',
    'id'      => 'wiki_test_colorpicker',
    'type'    => 'colorpicker',
    'default' => '#ffffff',
) );
```

CSS Field Class:

```
cmb-type-colorpicker
```

checkbox

Standard checkbox. Example:

```
$cmb->add_field( array(
    'name' => 'Test Checkbox',
    'desc' => 'field description (optional)',
    'id'   => 'wiki_test_checkbox',
    'type' => 'checkbox'
) );
```

CSS Field Class:

```
cmb-type-checkbox
```

multicheck and multicheck_inline

A field with multiple checkboxes (and multiple can be selected). Example:

```
$cmb->add_field( array(
    'name'    => 'Test Multi Checkbox',
    'desc'    => 'field description (optional)',
    'id'      => 'wiki_test_multicheckbox',
    'type'    => 'multicheck',
    'options' => array(
        'check1' => 'Check One',
        'check2' => 'Check Two',
        'check3' => 'Check Three',
```

```
    )
) );
```

CSS Field Class:

```
cmb-type-multicheck
```

Custom Field Attributes:

- `'select_all_button' => false` , Setting to false disables the 'Select All' button

radio

Standard radio buttons. Example:

```
$cmb->add_field( array(
    'name'             => 'Test Radio',
    'id'               => 'wiki_test_radio',
    'type'             => 'radio',
    'show_option_none' => true,
    'options'          => array(
        'standard' => __( 'Option One', 'cmb' ),
        'custom'   => __( 'Option Two', 'cmb' ),
        'none'     => __( 'Option Three', 'cmb' ),
    ),
) );
```

Set the optional paremter, `show_option_none` , to `true` to use the default text, 'None', or specify another value, i.e. 'No selection'. By default `show_option_none` is false.

CSS Field Class:

```
cmb-type-radio
```

radio_inline

Inline radio buttons. Example:

```
$cmb->add_field( array(
    'name'    => 'Test Radio inline',
    'id'      => 'wiki_test_radio_inline',
    'type'    => 'radio_inline',
    'options' => array(
        'standard' => __( 'Option One', 'cmb' ),
        'custom'   => __( 'Option Two', 'cmb' ),
        'none'     => __( 'Option Three', 'cmb' ),
    ),
) );
```

CSS Field Class:

```
cmb-type-radio-inline
```

select

Standard select dropdown. Example:

```
$cmb->add_field( array(
    'name'             => 'Test Select',
    'desc'             => 'Select an option',
    'id'               => 'wiki_test_select',
    'type'             => 'select',
    'show_option_none' => true,
    'default'          => 'custom',
    'options'          => array(
        'standard' => __( 'Option One', 'cmb' ),
        'custom'   => __( 'Option Two', 'cmb' ),
        'none'     => __( 'Option Three', 'cmb' ),
    ),
) );
```

Set the optional paremter, `show_option_none` , to `true` to use the default text, 'None', or specify another value, i.e. 'No selection'. By default `show_option_none` is false.

CSS Field Class:

```
cmb-type-select
```

Optional:

- All the types that take an `options` parameter can accept a callback. This callback will recieive the field object which you can use to check the object ID (`$field->object_id`). This can be handy if you need to build options based on the current post or context. The callback should return an array of options in the format displayed in these examples.

Example:

```
    // in the field array..
    'options' => 'show_cat_or_dog_options',


// Callback function
function show_cat_or_dog_options( $field ) {

    if ( has_tag( 'cats', $field->object_id ) ) {
        return array(
            'tabby'   => __( 'Tabby', 'cmb' ),
            'siamese' => __( 'Siamese', 'cmb' ),
            'calico'  => __( 'Calico', 'cmb' ),
        );
    } else {
        return array(
            'german-shepherd' => __( 'German Shepherd', 'cmb' ),
            'bulldog'         => __( 'Bulldog', 'cmb' ),
            'poodle'          => __( 'Poodle', 'cmb' ),
        );
    }
}
```

taxonomy_radio

Radio buttons pre-populated with taxonomy terms. Example:

```
$cmb->add_field( array(
    'name'     => 'Test Taxonomy Radio',
    'desc'     => 'Description Goes Here',
    'id'       => 'wiki_test_taxonomy_radio',
    'taxonomy' => '', // Enter Taxonomy Slug
    'type'     => 'taxonomy_radio',
    // Optional:
    'options' => array(
        'no_terms_text' => 'Sorry, no terms could be found.' // Change default text.
    ),
) );
```

CSS Field Class:

```
cmb-type-taxonomy-radio
```

taxonomy_radio_inline

Inline radio buttons pre-populated with taxonomy terms.

CSS Field Class:

```
cmb-type-taxonomy-radio-inline
```

taxonomy_select

A select field pre-populated with taxonomy terms. Example:

```
$cmb->add_field( array(
    'name'     => 'Test Taxonomy Select',
    'desc'     => 'Description Goes Here',
    'id'       => 'wiki_test_taxonomy_select',
    'taxonomy' => 'category', //Enter Taxonomy Slug
    'type'     => 'taxonomy_select',
) );
```

CSS Field Class:

```
cmb-type-taxonomy-select
```

taxonomy_multicheck

A field with checkboxes with taxonomy terms, and multiple terms can be selected

```
$cmb->add_field( array(
    'name'     => 'Test Taxonomy Multicheck',
    'desc'     => 'Description Goes Here',
    'id'       => 'wiki_test_taxonomy_multicheck',
    'taxonomy' => '', //Enter Taxonomy Slug
    'type'     => 'taxonomy_multicheck',
    // Optional:
    'options' => array(
        'no_terms_text' => 'Sorry, no terms could be found.' // Change default text.
    ),
```

```
) );
```

CSS Field Class:

```
cmb-type-taxonomy-multicheck
```

Custom Field Attributes:

- `'select_all_button' => false` , Setting to false disables the 'Select All' button

`taxonomy_multicheck_inline`

Inline checkboxes with taxonomy terms.

CSS Field Class:

```
cmb-type-taxonomy-multicheck-inline
```

Custom Field Attributes:

- `'select_all_button' => false` , Setting to false disables the 'Select All' button

Notes

To retrieve the values from the taxonomy fields, use `get_the_terms` , not `get_post_meta` , etc. The taxonomy fields are not intended to provide an arbitrary list of terms to pick from, but are intended to be a replacement for the default taxonomy meta-boxes. I.e. they are meant to set the taxonomy terms on an object, and will not save as a meta value. Any other use of these types will be hacky and/or buggy. Suggest looking at building a custom field type instead - Example.

`wysiwyg`

A metabox with TinyMCE editor (same as WordPress' visual editor). Example:

```
$cmb->add_field( array(
    'name'    => 'Test wysiwyg',
    'desc'    => 'field description (optional)',
    'id'      => 'wiki_test_wysiwyg',
    'type'    => 'wysiwyg',
    'options' => array(),
) );
```

CSS Field Class:

```
cmb-type-wysiwyg
```

Notes

Text added in a wysiwyg field will not have paragraph tags automatically added, the same is true of standard WordPress post content editing with the WYSIWYG. When outputting formatted text, wrap your get_post_meta() call with wpautop to generate the paragraph tags.

```
<?php echo wpautop( get_post_meta( get_the_ID(), 'wiki_test_wysiwyg', true ) ); ?>
```

If you want oembed filters to apply to the wysiwyg content, add this helper function to your theme

or plugin:

```
function yourprefix_get_wysiwyg_output( $meta_key, $post_id = 0 ) {
    global $wp_embed;

    $post_id = $post_id ? $post_id : get_the_id();

    $content = get_post_meta( $post_id, $meta_key, 1 );
    $content = $wp_embed->autoembed( $content );
    $content = $wp_embed->run_shortcode( $content );
    $content = do_shortcode( $content );
    $content = wpautop( $content );

    return $content;
}

...

echo yourprefix_get_wysiwyg_output( 'wiki_test_wysiwyg', get_the_ID() );
```

The options array allows you to customize the settings of the wysiwyg. Here's an example with all the options:

```
array(
    'name'    => 'Test wysiwyg',
    'desc'    => 'field description (optional)',
    'id'      => 'wiki_test_wysiwyg',
    'type'    => 'wysiwyg',
    'options' => array(
        'wpautop' => true, // use wpautop?
        'media_buttons' => true, // show insert/upload button(s)
        'textarea_name' => $editor_id, // set the textarea name to something differe
        'textarea_rows' => get_option('default_post_edit_rows', 10), // rows="..."
        'tabindex' => '',
        'editor_css' => '', // intended for extra styles for both visual and HTML ed
        'editor_class' => '', // add extra class(es) to the editor textarea
        'teeny' => false, // output the minimal editor config used in Press This
        'dfw' => false, // replace the default fullscreen with DFW (needs specific c
        'tinymce' => true, // load TinyMCE, can be used to pass settings directly to
        'quicktags' => true // load Quicktags, can be used to pass settings directly
    ),
),
```

The `'id'` should not be set to 'content' as the standard editor has this id and it will result in a non working editor.

file

A file uploader. By default it will store the file url and allow either attachments or URLs. This field type will also store the attachment ID (useful for getting different image sizes). It will store it in `$id . '_id'`, so if your field id is `_wiki_test_image` the ID is stored in `_wiki_test_image_id`. You can also limit it to only allowing attachments (can't manually type in a URL), which is also useful if you plan to use the attachment ID. The example shows its default values, with possible values commented inline. Example:

```
$cmb->add_field( array(
```

```
    'name'    => 'Test File',
    'desc'    => 'Upload an image or enter an URL.',
    'id'      => 'wiki_test_image',
    'type'    => 'file',
    // Optional:
    'options' => array(
        'url' => false, // Hide the text input for the url
        'add_upload_file_text' => 'Add File' // Change upload button text. Default:
    ),
) );
```

CSS Field Class:

```
cmb-type-file
```

Example using the `_wiki_test_image_id` to retrieve a medium image:

```
$image = wp_get_attachment_image( get_post_meta( get_the_ID(), 'wiki_test_image_id',
```

file_list

A file uploader that allows you to add as many files as you want. This is a repeatable field, and will store its data in an array, with the attachment ID as the array key and the attachment url as the value. Example:

```
$cmb->add_field( array(
    'name' => 'Test File List',
    'desc' => '',
    'id'   => 'wiki_test_file_list',
    'type' => 'file_list',
    // 'preview_size' => array( 100, 100 ), // Default: array( 50, 50 )
    // Optional, override default text strings
    'options' => array(
        'add_upload_files_text' => 'Replacement', // default: "Add or Upload Files"
        'remove_image_text' => 'Replacement', // default: "Remove Image"
        'file_text' => 'Replacement', // default: "File:"
        'file_download_text' => 'Replacement', // default: "Download"
        'remove_text' => 'Replacement', // default: "Remove"
    ),
) );
```

CSS Field Class:

```
cmb-type-file-list
```

Custom Field Attributes:

- `preview_size` Changes the size of the preview images in the field. Default: array(50, 50).

Sample function for getting and outputting `file_list` images

```
/**
 * Sample template tag function for outputting a cmb2 file_list
 *
 * @param  string  $file_list_meta_key The field meta key. ('wiki_test_file_list')
 * @param  string  $img_size           Size of image to show
 */
function cmb2_output_file_list( $file_list_meta_key, $img_size = 'medium' ) {
```

```
    // Get the list of files
    $files = get_post_meta( get_the_ID(), $file_list_meta_key, 1 );

    echo '<div class="file-list-wrap">';
    // Loop through them and output an image
    foreach ( (array) $files as $attachment_id => $attachment_url ) {
        echo '<div class="file-list-image">';
        echo wp_get_attachment_image( $attachment_id, $img_size );
        echo '</div>';
    }
    echo '</div>';
}
```

To use in your template (in the loop):

```
<?php cmb2_output_file_list( 'wiki_test_file_list', 'small' ); ?>
```

oembed

Displays embedded media inline using WordPress' built-in oEmbed support. See codex.wordpress.org/Embeds for more info and for a list of embed services supported. (added in 0.9.1)

```
$cmb->add_field( array(
    'name' => 'oEmbed',
    'desc' => 'Enter a youtube, twitter, or instagram URL. Supports services listed
    'id'   => 'wiki_test_embed',
    'type' => 'oembed',
) );
```

CSS Field Class:

```
cmb-type-oembed
```

Notes

Text added in a `oembed` field will not automatically display the embed in your theme. To generate the embed in your theme, this is a method you could use:

```
$url = esc_url( get_post_meta( get_the_ID(), 'wiki_test_embed', 1 ) );
echo wp_oembed_get( $url );
```

group

Hybrid field that supports adding other fields as a repeatable group. Example:

```
$group_field_id = $cmb->add_field( array(
    'id'          => 'wiki_test_repeat_group',
    'type'        => 'group',
    'description' => __( 'Generates reusable form entries', 'cmb' ),
        'repeatable'  => true, // use false if you want non-repeatable group
    'options'     => array(
```

```
        'group_title'   => __( 'Entry {#}', 'cmb' ), // since version 1.1.4, {#} get
        'add_button'    => __( 'Add Another Entry', 'cmb' ),
        'remove_button' => __( 'Remove Entry', 'cmb' ),
        'sortable'      => true, // beta
        // 'closed'     => true, // true to have the groups closed by default
    ),
) );

// Id's for group's fields only need to be unique for the group. Prefix is not neede
$cmb->add_group_field( $group_field_id, array(
    'name' => 'Entry Title',
    'id'   => 'title',
    'type' => 'text',
    // 'repeatable' => true, // Repeatable fields are supported w/in repeatable grou
) );

$cmb->add_group_field( $group_field_id, array(
    'name' => 'Description',
    'description' => 'Write a short description for this entry',
    'id'   => 'description',
    'type' => 'textarea_small',
) );

$cmb->add_group_field( $group_field_id, array(
    'name' => 'Entry Image',
    'id'   => 'image',
    'type' => 'file',
) );

$cmb->add_group_field( $group_field_id, array(
    'name' => 'Image Caption',
    'id'   => 'image_caption',
    'type' => 'text',
) );
```

CSS Field Class:

```
cmb-field-list
```

All repeatable group entries will be saved as an array to that meta-key. Example usage to pull data back:

```
$entries = get_post_meta( get_the_ID(), 'wiki_test_repeat_group', true );

foreach ( (array) $entries as $key => $entry ) {

    $img = $title = $desc = $caption = '';

    if ( isset( $entry['title'] ) )
        $title = esc_html( $entry['title'] );

    if ( isset( $entry['description'] ) )
        $desc = wpautop( $entry['description'] );

    if ( isset( $entry['image_id'] ) ) {
        $img = wp_get_attachment_image( $entry['image_id'], 'share-pick', null, arra
            'class' => 'thumb',
        ) );
    }
    $caption = isset( $entry['image_caption'] ) ? wpautop( $entry['image_caption'] )

    // Do something with the data
}
```

Custom Field Types

You can define your own field types as well.

Common Field Parameters

Most (if not all) fields support these parameters:

- `name` : The field label
- `desc` : Field description. Usually under or adjacent to the field input.
- `id` : The data key. If using for posts, will be the post-meta key. If using for an options page, will be the array key.
- `type` : What makes the whole thing work.
- `repeatable` : Supported by most, and will make the individual field a repeatable one.
- `default` : Specify a default value for the field.
- `show_names` : Hide label for the field.
- `options` : For fields that take an options array. These include: `select` , `radio` , `multicheck` , `wysiwyg` and `group` . Can also accept a callback. The callback will receive the CMB2_Field `$field` object as an argument.
- `before` , `after` , `before_row` , `after_row` , `before_field` , `after_field` : These allow you to add arbitrary text/markup at different points in the field markup. These also accept a callback. The callback will receive `$field_args` as the first argument, and the CMB2_Field `$field` object as the second argument.
- `row_classes` : This parameter allows you to add additional classes to the cmb-row wrap. This parameter can take a string, or array, or can take a callback that returns a string or array. Like above, the callback will receive `$field_args` as the first argument, and the CMB2_Field `$field` object as the second argument.
- `on_front` : If you're planning on using your metabox fields on the front-end as well (user-facing), then you can specify that certain fields do not get displayed there by setting this parameter to `false` .

- `attributes` : Will modify default attributes (class, input type, rows, etc), or add your own (placeholder, data attributes). Example:

```
$cmb->add_field( array(
    'name'        => 'Extra Small Textarea',
    'id'          => 'wiki_test_xtra_small_textarea',
    'type'        => 'textarea_small',
    'attributes'  => array(
        'placeholder' => 'A small amount of text',
        'rows'        => 3,
        'required'    => 'required',
    ),
) );
```

- `show_on_cb` : A callback to conditionally display a field. Callback function should return a boolean (true/false) value. Function passes in the current field object. Example:

```
$cmb->add_field( array(
    'name'       => __( 'Test Text', 'cmb' ),
    'id'         => 'wiki_test_text',
    'type'       => 'text',
    'show_on_cb' => 'cmb_only_show_for_user_1', // function should return a bool
```

```
) );

...

/**
 * Only display a field if the current user is 1
 * @param  object $field Current field object
 * @return bool          True if current user's ID is 1
 */
function cmb_only_show_for_user_1( $field ) {
    // Returns true if current user's ID is 1, else false
    return 1 === get_current_user_id();
}
```

- `escape_cb` : Bypass the CMB escaping (escapes before display) methods with your own callback. Set to `false` if you do not want any escaping (not recommended).
- `sanitization_cb` : Bypass the CMB sanitization (sanitizes before saving) methods with your own callback. Set to `false` if you do not want any sanitization (not recommended).
- `render_row_cb` : Bypass the CMB row rendering. You will completely responsible for outputting that row's html. The callback function gets passed the field `$args` array, and the `$field` object.

© 2015 GitHub, Inc. Terms Privacy Security Contact Help

Status API Training Shop Blog About Pricing