

Adding custom javascript to fields



- [Getting Started](#)
 - [Include](#)
 - [functions.php](#)
 - [Inline](#)
 - [functions.php](#)
- [Actions](#)
 - [Ready](#)
 - [Append](#)
 - [Load](#)
 - [Remove](#)
 - [sortstart](#)
 - [sortstop](#)
 - [hide_field](#)
 - [show_field](#)
 - [date_picker_init](#)
 - [date_time_picker_init](#)
 - [time_picker_init](#)
 - [select2_init](#)
 - [wysiwyg_tinymce_init](#)
 - [wysiwyg_quicktags_init](#)
 - [google_map_init](#)
- [Filters](#)
 - [validation_complete](#)
 - [wysiwyg_tinymce_settings](#)
 - [wysiwyg_quicktags_settings](#)
 - [date_picker_args](#)

- [date_time_picker_args](#)
 - [time_picker_args](#)
 - [google_map_args](#)
 - [select2_args](#)
 - [select2_ajax_data](#)
 - [color_picker_args](#)
-



☐ We've launched a new JavaScript API full of powerful functions, actions and filters! If you are using ACF 5.7 or above, please checkout our [JavaScript API](#).

This article will cover how to add custom JS to interact with and modify ACF fields and settings. Similar to the many WordPress actions and filters available in PHP, ACF adopts a similar model for it's Javascript.

Getting Started

There are two methods to add custom javascript to the WordPress dashboard; include a script file or append an inline script. It is recommended that you include a JS file if you intend to write lots of functionality.

Include

To include a JS file within the dashboard, you can make use of the `wp_enqueue_script` function like so.

functions.php

```
function my_admin_enqueue_scripts() {  
  
    wp_enqueue_script( 'my-admin-js', get_template_directory_uri() . '/js/example.js',  
    array(), '1.0.0', true );  
  
}
```

```
add_action('acf/input/admin_enqueue_scripts', 'my_admin_enqueue_scripts');
```

Inline

To append inline JS within the dashboard, you can make use of the `acf/input/admin_footer` action. This action is run in the footer of any admin page where ACF fields may exist.

functions.php

```
function my_acf_input_admin_footer() {

    ?>
    <script type="text/javascript">
    (function($) {

        // JS here

    })(jQuery);
    </script>
    <?php

}

add_action('acf/input/admin_footer', 'my_acf_input_admin_footer');
```

Actions

Below is a list of all available JS actions. Hooking into an action is made possible by a JS function called `add_action` on the `acf` object. If you are using the **ready** action, it is a good idea to also hook into the **append** action with the same functionality.

Ready

Called when the document is ready once the page has initially loaded.

```
acf.add_action('ready', function( $el ){

    // $el will be equivalent to $('body')
```

```
// find a specific field
var $field = $('#my-wrapper-id');

// do something to $field

});
```

This action also fires on each field via the actions `ready_field` and `ready_field/type={$field_type}`.

Append

Called when new HTML is appended to the page. This occurs when adding a new repeater row, adding a new flexible content layout and when new field groups are loaded via AJAX. If you are using the append action, it is a good idea to also hook into this action with the same functionality.

```
acf.add_action('append', function( $el ){

    // $el will be equivalent to the new element being appended $('tr.row')

    // find a specific field
    var $field = $el.find('#my-wrapper-id');

    // do something to $field

});
```

This action also fires on each field via the actions `append_field` and `append_field/type={$field_type}`.

Load

Called when the window has loaded all assets. This action is desired over 'ready' when the width and height if an image is needed for logic.

```
acf.add_action('load', function( $el ){

    // $el will be equivalent to $('body')

    // find a specific field
    var $field = $el.find('#my-wrapper-id');

    // do something to $field

});
```

This action also fires on each field via the actions `load_field` and `load_field/type={$field_type}`.

Remove

Called when HTML is removed from the page. This occurs when removing a repeater row or removing a flexible content layout.

```
acf.add_action('remove', function( $el ){

    // $el will be equivalent to the new element being removed $('tr.row')

    // find a specific field
    var $field = $el.find('#my-wrapper-id');

    // do something to $field

});
```

This action also fires on each field via the actions `remove_field` and `remove_field/type={$field_type}`.

sortstart

Called when HTML has begun to move around the page via drag/drop. This occurs when reordering a repeater row or reordering a flexible content layout.

```
acf.add_action('sortstart', function( $el ){

    // $el will be equivalent to the new element being moved $('tr.row')

    // find a specific field
    var $field = $el.find('#my-wrapper-id');

    // do something to $field

});
```

This action also fires on each field via the actions `sortstart_field` and `sortstart_field/type={$field_type}`.

sortstop

Called when HTML has finished being moved around the page via drag/drop. This occurs when reordering a repeater row or reordering a flexible content layout.

```
acf.add_action('sortstop', function( $el ){

    // $el will be equivalent to the new element being moved $('tr.row')

    // find a specific field
    var $field = $el.find('#my-wrapper-id');

    // do something to $field

});
```

This action also fires on each field via the actions `sortstop_field` and `sortstop_field/type={$field_type}`.

hide_field

Called when a field has been hidden via either a tab or by conditional logic.

```
acf.add_action('hide_field', function( $field, context ){

    // context is a string of either 'tab' or 'conditional_logic'

    // do something to $field

});
```

This action also fires on a field specific basis `hide_field/type={$field_type}`.

show_field

Called when a field has been shown via either a tab or by conditional logic.

```
acf.add_action('show_field', function( $field, context ){

    // context is a string of either 'tab' or 'conditional_logic'

    // do something to $field

});
```

This action also fires on a field specific basis `show_field/type={$field_type}`.

date_picker_init

Called when a date picker has been initialized. Added in v5.5.8

```
acf.add_action('date_picker_init', function( $input, args, $field ){

    // $input (jQuery) text input element
    // args (object) args given to the datepicker function
    // $field (jQuery) field element

});
```

date_time_picker_init

Called when a date time picker has been initialized. Added in v5.5.8

```
acf.add_action('date_time_picker_init', function( $input, args, $field ){

    // $input (jQuery) text input element
    // args (object) args given to the datepicker function
    // $field (jQuery) field element

});
```

time_picker_init

Called when a time picker has been initialized. Added in v5.5.8

```
acf.add_action('time_picker_init', function( $input, args, $field ){

    // $input (jQuery) text input element
    // args (object) args given to the datepicker function
    // $field (jQuery) field element

});
```

select2_init

Called when a Select2 element has been initialized. By default, ACF includes the Select2 v3 library which requires a hidden input to be used for Select2 initialization. It is possible to include the Select2 v4 library, and if so, the \$input arg will supply a 'select' element instead of a hidden input due to changes in the library. Added in v5.5.8

```
acf.add_action('select2_init', function( $input, args, settings, $field ){

    // $input (jQuery) hidden input element
    // args (object) args given to the select2 function
    // settings (object) settings given to the acf.select2 function
    // $field (jQuery) field element

});
```

wysiwyg_tinymce_init

Called when a WYSIWYG tinymce element has been initialized. Added in v5.5.8

```
acf.add_action('wysiwyg_tinymce_init', function( ed, id, mceInit, $field ){

    // ed (object) tinymce object returned by the init function
    // id (string) identifier for the tinymce instance
    // mceInit (object) args given to the tinymce function
    // $field (jQuery) field element

});
```

wysiwyg_quicktags_init

Called when a WYSIWYG quicktags element has been initialized. Each tinymce instance can also contain a 'text' edit mode that shows basic 'quicktag' buttons. Added in v5.5.8

```
acf.add_action('wysiwyg_quicktags_init', function( qtag, id, qtInit, $field ){

    // qtag (object) quick tag object returned by the init function
    // id (string) identifier for the qtag instance
    // qtInit (object) args given to the quick tag function
    // $field (jQuery) field element

});
```

google_map_init

Called when a Google Map element has been initialized. Added in v5.5.11

```
acf.add_action('google_map_init', function( map, marker, $field ){

    // map (object) google map object returned by the google.maps.Map() function
    // marker (object) marker object returned by the google.maps.Marker() function
    // $field (jQuery) field element

});
```

Filters

validation_complete

This filter allows the validation JSON response to be customized. Called after AJAX validation is complete but before errors are shown or the form is submitted.

```
acf.add_filter('validation_complete', function( json, $form ){

    // if errors?
    if( json.errors ) {

    }

    // return
    return json;

});
```

wysiwyg_tinymce_settings

This filter allows the tinyMCE settings to be customized for each WYSIWYG field. Called before the tinyMCE instance is created.

```
acf.add_filter('wysiwyg_tinymce_settings', function( mceInit, id, $field ){

    // do something to mceInit

    // return
    return mceInit;

});
```

wysiwyg_quicktags_settings

This filter allows the quicktags settings to be customized for each WYSIWYG field. Called before the text instance is created.

```
acf.add_filter('wysiwyg_quicktags_settings', function( qtInit, id, $field ){

    // do something to qtInit

    // return
    return qtInit;

});
```

date_picker_args

This filter allows the date picker settings to be customized for each date picker field. Called before the date picker instance is created. A full list of settings can be found here:

<https://api.jqueryui.com/datepicker/>

```
acf.add_filter('date_picker_args', function( args, $field ){

    // do something to args

    // return
    return args;

});
```

date_time_picker_args

This filter allows the date time picker settings to be customized for each date time picker field. Called before the date time picker instance is created. A full list of settings can be found here:

<http://trentrichardson.com/examples/timepicker/>

```
acf.add_filter('date_time_picker_args', function( args, $field ){

    // do something to args


    // return
    return args;

});
```

time_picker_args

This filter allows the time picker settings to be customized for each time picker field. Called before the time picker instance is created. A full list of settings can be found here:

<http://trentrichardson.com/examples/timepicker/>

```
acf.add_filter('time_picker_args', function( args, $field ){

    // do something to args


    // return
    return args;

});
```

google_map_args

This filter allows the google maps settings to be customized for each google maps field. Called before the map instance is created.

```
acf.add_filter('google_map_args', function( args, $field ){

    // do something to args


    // return
    return args;

});
```

```
});
```

select2_args

This filter allows the select2 settings to be customized for each select field. Called before the select2 instance is created.

```
acf.add_filter('select2_args', function( args, $select, settings, $field ){

    // do something to args

    // return
    return args;

});
```

select2_ajax_data

This filter allows the data sent in an ajax request to be customized for each select field. Called before the select2 ajax request is created.

```
acf.add_filter('select2_ajax_data', function( data, args, $input, $field ){

    // do something to data

    // return
    return data;

});
```

color_picker_args

This filter allows the color picker (wpColorPicker) settings to be customized for each color picker field. This filter is called before the wpColorPicker instance is created. Added in v 5.3.6

```
acf.add_filter('color_picker_args', function( args, $field ){

    // do something to args
    args.palettes = ['#5ee8bf', '#2f353e', '#f55e4f']

    // return
    return args;

});
```

```
### #2
[] [] [] 2026-08-29 15:44:03 UTC
[] [] [] 2026-08-29 15:44:03 UTC
```