

```
add_action('wordcamp_jkl',  
    'johdatus_filttereihin');
```

WordCamp JKL 9.2.2018

Mikko Saari | Painava sana oy
mikko@mikkosaari.fi | @msaari

Mitä filterit ja actionit ovat?

- Filter ja action ovat melkein samoja asioita: molemmat ovat WordPressin toiminnan osaksi koukutettuja omia funktioita.
- Filter saa jonkun arvon, jota se tarkastelee tai muuttaa.
- Action on tilaisuus tehdä jotain osana prosessia.
- Kummatkin edellyttävät koukun, eli koodissa pitää olla filterille tai actionille varattu paikka.

```
$filter_data = array($hits, $q);  
$hits_filters_applied = apply_filters('relevanssi_hits_filter', $filter_data);  
$hits = array_values($hits_filters_applied[0]);
```

```
/**  
 * Filters the array of retrieved posts after they've been fetched and  
 * internally processed.  
 *  
 * @since 1.5.0  
 *  
 * @param array $posts The array of retrieved posts.  
 * @param WP_Query $this The WP_Query instance (passed by reference).  
 */  
$this->posts = apply_filters_ref_array( 'the_posts', array( $this->posts, &$this ) );  
}
```

```
/**  
 * After the roles have been initialized, allow plugins to add their own roles.  
 *  
 * @since 4.7.0  
 *  
 * @param WP_Roles $this A reference to the WP_Roles object.  
 */  
do_action( 'wp_roles_init', $this );
```

Mitä hyötyä?

- Tapa muuttaa WordPressin tai pluginin toimintaa turvallisesti ja kestävästi.
- Suora muutos WordPressin tai pluginin tiedostoihin jyrätään seuraavassa päivityksessä.
- Ulkopuolelta filtterin tai actionin kautta tehty muutos kestää (yleensä) päivittämisen.

Esimerkkejä

TablePress-pluginilla tehtävässä taulukossa otsikon pitäisi olla `<h3>`, mutta etusivulla `<h2>`. Ei onnistu asetuksista, mutta onnistuu näin:

```
add_filter('tablepress_print_name_html_tag', 'table_h2_to_h3');  
function table_h2_to_h3($tag) {  
    if (is_front_page()) return "h2";  
    return "h3";  
}
```

Esimerkkejä

Entäs jos taulukko on desktopilla hyvä, mutta mobiilipuolella pitäisi oikeastaan saada samasta datasta pyöräytettyä `<div>`-pohjainen ratkaisu?
Pistetään filterillä koko taulu uusiksi!

```
add_filter('tablepress_table_output', 'mobiilitaulut', 10, 2);
function mobiilitaulut($output, $table) {
    if (function_exists('is_mobile') && is_mobile()) {
        $class = $table['options']['extra_css_classes'];

        $mobiilitaulu = '<div class="mobiilitaulu">';
        foreach ($table['data'] as $row) {
```

Esimerkkejä

Entäs jos Relevanssin indeksiin halutaan kaikki artikkelit, paitsi post_typessä "uutinen" olevat artikkelit, jotka on kirjoittanut käyttäjä numero 3 ja jotka käsittelevät jalkapalloa?

```
add_filter('relevanssi_do_not_index', 'indeksiesto', 10, 2);
function indeksiesto($ohita, $post_id) {
    $post = get_post($post_id);
    if ($post->post_type == "uutinen") {
        if ($post->post_author == 3) {
            if (has_term("jalkapallo", "post_tag", $post_id)) $ohita = true;
        }
    }
    return $ohita;
}
```

Esimerkkejä

Ärsyttääkö kun artikkelin muokkaussivulla avainkenttien listassa ei näy kaikkia avainkenttiä, jos niitä on liikaa?

```
add_filter( 'postmeta_form_limit' , 'postmeta_form_custom_limit' );  
function postmeta_form_custom_limit( $limit ) {  
    $limit = 1000;  
    return $limit;  
}
```

```
add_action('admin_menu', 'relevanssi_menu');
add_filter('the_posts', 'relevanssi_query', 99, 2);
add_action('delete_post', 'relevanssi_delete');
add_action('comment_post', 'relevanssi_comment_index');
add_action('edit_comment', 'relevanssi_comment_edit');
add_action('delete_comment', 'relevanssi_comment_remove');
add_action('wp_insert_post', 'relevanssi_insert_edit', 99, 1 );
add_action('delete_attachment', 'relevanssi_delete');
add_action('add_attachment', 'relevanssi_publish', 12);
add_action('edit_attachment', 'relevanssi_edit');
add_action('transition_post_status', 'relevanssi_update_child_posts', 99, 3);
add_action('init', 'relevanssi_init');
add_action('relevanssi_trim_logs', 'relevanssi_trim_logs');
add_filter('relevanssi_hits_filter', 'relevanssi_wpml_filter');
add_filter('relevanssi_modify_wp_query', 'relevanssi_polylang_filter');
add_filter('posts_request', 'relevanssi_prevent_default_request', 10, 2 );
add_filter('relevanssi_remove_punctuation', 'relevanssi_remove_punct');
add_filter('relevanssi_post_ok', 'relevanssi_default_post_ok', 9, 2);
add_filter('relevanssi_query_filter', 'relevanssi_limit_filter');
add_filter('query_vars', 'relevanssi_query_vars');
add_filter('relevanssi_indexing_values', 'relevanssi_update_doc_count', 98, 2);
add_filter('relevanssi_pre_excerpt_content', 'relevanssi_remove_page_builder_shortcodes',
add_filter('rest_api_init', 'relevanssi_rest_api_disable');
```

Filtterien käyttäminen

- `apply_filters('koukun_nimi', suodatettava arvo, parametrit)`
 - Parametrejä voi olla useita.
- `add_filter('koukun_nimi', 'funktion_nimi', prioriteetti, parametrien lukumäärä)`
 - Prioriteetti määrittelee järjestyksen, jossa samaan koukkuun asetetut filtterit ajetaan.
 - Toinen numero kertoo filtterin parametrien lukumäärän.
 - Funktion pitää palauttaa lopuksi `suodatettu arvo`.

Actionien käyttäminen

- `do_action('koukun_nimi', parametrit)`
 - Tässäkin voi olla useita parametrejä, mutta mitään niistä ei tarvitse eikä voi palauttaa.
- `add_action('koukun_nimi', 'funktion_nimi', prioriteetti, parametrien lukumäärä)`
 - Samoin tässäkin prioriteetti ja parametrien määrä.
 - Tämä on itse asiassa tismalleen sama funktio kuin `add_filter()`, ei ole väliä kumpaa käytät.

apply_filters_ref_array()?

- Tällaistaakin muotoa näkee:
 - `$args = array('foo', 'bar', 'baz');`
`apply_filters_ref_array('koukku', $args);`
- Tämä on sama kuin:
 - `apply_filters('koukku', 'foo', 'bar', 'baz');`
- Arrayn ensimmäinen elementti on suodatettava arvo, muut lisäparametrejä.
- Aikaisemmin parametri oli viittaus, nykyään (PHP 5.4+) ei yleensä enää ole.
- Actioneille on vastaava `do_action_ref_array()`

Minne oma koodi laitetaan?

- Tavallisin paikka on teeman functions.php-tiedosto, joka on olemassa tällaista varten.
- Jos käytät teemaa, jota päivitetään, päivitys jyrää muutokset, eli silloin oikea paikka on lapsiteeman functions.php.
- Jos muutos ei nimenomaisesti liity teemaan vaan sitin toiminnallisuuteen muuten, paras tapa on tehdä oma miniplugini koodimuutoksia varten: silloin teeman vaihtaminen ei riko kaikkea toiminnallisuutta.

Miniplugini

- ```
<?php
/*
Plugin Name: Oma toiminnallisuus
Description: Omat funktiot tänne!
Version: 1
*/

add_filter('the_content', 'sisältösuodatin');
function sisältösuodatin($sisältö) {
 ...
 return $sisältö;
}
```

# Koukut teemoissa

- Koukut eivät ole vain WordPressin ja lisäosien juttu, niitä voi käyttää myös teemoissa.
- Teeman rakentamisessa tarvitaan paljon koukkuja, joihin teeman ominaisuuksia liitetään.
- Genesis ja Thesis tunnettuja esimerkkejä koukkuihin perustuvista teemoista: muutoksia ei tehdä suoraan sivupohjiin, vaan koukuilla.
- Hyvä idea teoriassa, käytännössä ei toimi ihan yhtä hyvin: teemaa voi silloin muokata vain sen verran kuin koukut antavat periksi.

# Koukkujen tarkastelua

- `has_filter('koukun_nimi', 'funktion_nimi')`
  - Kertoo, onko tietty funktio tietyssä koukussa. Vastaavasti `has_action()`.
- `current_filter()`
  - Kertoo, minkä filterikoukun suoritus on menossa.
- `did_action('koukku')`
  - Kertoo kuinka monta kertaa koukku on suoritettu. Sama koukku voi lauaa monta kertaa, joten jos halutaan varmistaa, että joku asia tehdään vain ensimmäisellä kerralla, tällä voi varmistaa sen.

# Yleistyökalu

- `global $wp_filter;`
  - Sisältää kaikki rekisteröidyt filtterit ja actionit.
- `var_dump($wp_filter['koukun_nimi']);`
  - Näyttää kaikki koukuun rekisteröidyt funktiot.

# Funktioiden poistamista

- `remove_action('koukku', 'funktio', prioriteetti)`  
ja `remove_filter('koukku', 'funktio', prioriteetti)`
  - Poistaa filtterin tai actionin, prioriteettiä myöden pitää olla yksityiskohdat samat kuin asetettaessa.
- `remove_all_filters('koukku', prioriteetti)` ja `remove_all_actions('koukku', prioriteetti)`
  - Pyyhkäisee kaikki annetun prioriteetin funktiot koustusta. Jos prioriteettia ei käytä, poistetaan kaikki.

# Varo ikuista silmukkaa

- Actioneiden kohdalla pitää joissain tapauksissa varoa ikuiseen silmukkaan joutumista.
- Jos actionin sisältä kutsuu funktiota, joka ajaa saman actionin, silmukka on valmis!
- Esimerkki: koukussa `save_post` tekee muutoksia artikkeliin ja tallentaa ne `wp_update_post()` -funktiolla, joka suorittaa lopuksi `save_post`-koukun.
- ```
remove_action('save_post', 'oma_funktio' );  
wp_update_post($parametrit);  
add_action('save_post', 'oma_funktio');
```

Miten koukut löytyvät?

- Katso Codexista ("Filter Reference", "Action Reference").
- Pluginien kohdalla pitää sukeltaa lähdekoodiin, jos kehittäjä ei ole ystävällisesti dokumentoinut filttäreitään.
- Jos pluginissa ei ole valmiiksi sopivaa koukkua, pyydä kehittäjää lisäämään se! Tämä on yleensä helppo juttu.

Relevanssi has lots of filter hooks you can use to modify how the plugin behaves. Most of these are pretty advanced and an average user probably doesn't have to worry about them, but if you're going to do something more demanding, knowledge of these filters will help.

If you do something that would be made easier by a new filter, please let me know – I do like adding new filter hooks, as that is a very easy way to add flexibility to the plugin with little effort.

relevanssi_accents_replacement_arrays (\$replacement_array)

Relevanssi uses `str_replace()` to replace accented characters with their non-accented variants in order to make excerpts and highlighting match both forms. This filter can be used to adjust how this replacement is done. The parameter array has two parts: "from" has the non-accented forms and "to" has the matching accented letters.

relevanssi_admin_search_ok (true, \$query)

This filter can modify whether the default admin search query by WP is blocked or not.

relevanssi_allow_one_letter_highlights (false)

If you add a filter hook here that returns *true*, Relevanssi highlighting will highlight one-letter

Plugin API/Filter Reference

Languages: **English** • [日本語](#) • [\(Add your language\)](#)

This article contains an extensive (but not 100% comprehensive) list of the filter hooks available for use in plugin development in Version 2.1 and above of WordPress. For more information:

- To learn more about what filter and action hooks are, see [Plugin API](#).
- To learn about writing plugins in general, see [Writing a Plugin](#).
- For a reference list of action hooks, see [Plugin API/Action Reference](#).
- For information about filter and action hooks in previous versions of WordPress, see [Plugin API/Hooks 2.0.x](#).
- For an automatically-generated list of *all* WordPress hooks, see the [WordPress Hooks Database](#)

Note: If you want to add to or clarify this documentation, please follow the style of the existing entries. Describe what data the filter is applied to, and if the filter function takes additional arguments, describe the argument list.

Post, Page, and Attachment (Upload) Filters

See also [#Category and Term Filters](#), [#Author and User Filters](#), [#Link Filters](#), [#Date and Time Filters](#), and [#Administrative Filters](#) below.

Database Reads

Filters in this section are applied to information read from the database, prior to displaying on a page or editing screen.

`attachment_fields_to_edit`

applied to the form fields to be displayed when editing an attachment. Called in the `get_attachment_fields_to_edit` function. Filter function arguments: an array of form fields, the post object.

Contents

- [1 Post, Page, and Attachment \(Upload\) Filters](#)
 - [1.1 Database Reads](#)
 - [1.2 Database Writes](#)
- [2 Comment, Trackback, and Ping Filters](#)
 - [2.1 Database Reads](#)
 - [2.2 Database Writes](#)
- [3 Category and Term Filters](#)
 - [3.1 Database Reads](#)
 - [3.2 Database Writes](#)
- [4 Link Filters](#)

[Home Page](#)[WordPress Lessons](#)[Getting Started](#)[Working with WordPress](#)[Design and Layout](#)[Advanced Topics](#)[Troubleshooting](#)[Developer Docs](#)[About WordPress](#)

Codex Resources

[Community portal](#)[Current events](#)[Recent changes](#)[Random page](#)[Help](#)

Plugin API/Action Reference

Languages: **English** • [Русский](#) • [日本語](#) • [한국어](#) • [Português do Brasil](#) • ([Add your language](#))

This is a (hopefully) comprehensive list of action hooks available in WordPress version 2.1 and above. For more information:

- To learn more about what filter and action hooks are, see [Plugin API](#).
- To learn about writing plugins in general, see [Writing a Plugin](#).
- For a reference list of filter hooks, see [Plugin API/Filter Reference](#).
- For information about filter and action hooks in previous versions of WordPress, see [Plugin API/Hooks 2.0.x](#).

(If you want to add to or clarify this documentation, please follow the style of the existing entries. Describe when the action runs, and if the action function takes arguments, describe the arguments.)

Actions Run During a Typical Request

These actions are called when a logged-in user opens the home page in [Version 3.3.1](#). This list may show only the first time each action is called, and in many cases no function is hooked to the action. Themes and plugins can cause actions to be called multiple times and at differing times during a request. As proof of this, you can see action calls specific to the [Twenty Eleven](#) theme on this list. Cron tasks may also fire when a user visits the site, adding additional action calls. This list should be viewed as a guide line or approximation of WordPress action execution order, and not a concrete specification.

Actions are called with the function [do_action\(\)](#), except those marked (ref array), which are called with the function [do_action_ref_array\(\)](#).

Contents

- [1 Actions Run During a Typical Request](#)
- [2 Actions Run During an Admin Page Request](#)
- [3 Post, Page, Attachment, and Category Actions \(Admin\)](#)
 - [3.1 Taxonomy and Terms](#)
- [4 Comment, Ping, and Trackback Actions](#)
- [5 Blogroll Actions](#)
- [6 Feed Actions](#)
- [7 Template Actions](#)
- [8 Administrative Actions](#)
 - [8.1 Dashboard "Right Now" Widget Actions](#)
- [9 Advanced Actions](#)
- [10 Further Reading](#)

[Home Page](#)

[WordPress Lessons](#)

[Getting Started](#)

[Working with WordPress](#)

[Design and Layout](#)

[Advanced Topics](#)

[Troubleshooting](#)

[Developer Docs](#)

[About WordPress](#)

Codex Resources

[Community portal](#)

[Current events](#)

[Recent changes](#)

[Random page](#)

[Help](#)

- Action reference
-sivun kiinnostavinta
antia on lista
tavanomaisen
sivunlatauksen aikana
ajettavista actioneista.
- Lista on pitkä ja siinä
on monta hyödyllistä
actionia, joihin tarttua.
- Jotkut jutut on tehtävä
tietyssä vaiheessa
prosessia, jotta ne
toimivat oikein.

| | |
|--|---|
| <code>muplugins_loaded</code> | After must-use plugins are loaded. |
| <code>registered_taxonomy</code> | For category, post_tag, etc. |
| <code>registered_post_type</code> | For post, page, etc. |
| <code>plugins_loaded</code> | After active plugins and pluggable functions are loaded. |
| <code>sanitize_comment_cookies</code> | When comment cookies are sanitized. |
| <code>setup_theme</code> | Before the theme is loaded. |
| <code>load_textdomain</code> | For the default domain |
| <code>after_setup_theme</code> | Generally used to initialize theme settings/options. This is the first action hook available to themes , triggered immediately after the active theme's <i>functions.php</i> file is loaded. <code>add_theme_support()</code> should be called here, since the <code>init</code> action hook is too late to add some features. At this stage, the current user is not yet authenticated. |
| <code>auth_cookie_malformed</code> | |
| <code>auth_cookie_valid</code> | |
| <code>set_current_user</code> | |
| <code>init</code> | Typically used by plugins to initialize. The current user is already authenticated by this time. |
| ↳ <code>widgets_init</code> | Used to register sidebars. Fired at 'init' priority 1 (and so before 'init' actions with priority ≥ 1) |
| <code>register_sidebar</code> | For each sidebar and footer area |
| <code>wp_register_sidebar_widget</code> | For each widget |
| <code>wp_default_scripts</code> | (ref array) |
| <code>wp_default_styles</code> | (ref array) |
| <code>admin_bar_init</code> | |
| <code>add_admin_bar_menus</code> | |
| <code>wp_loaded</code> | After WordPress is fully loaded |
| <code>parse_request</code> | Allows manipulation of HTTP request handling (ref array) |
| <code>send_headers</code> | Allows customization of HTTP headers (ref array) |
| <code>parse_query</code> | After query variables are set (ref array) |
| <code>pre_get_posts</code> | Exposes the query-variables object before a query is executed. (ref array) |
| <code>posts_selection</code> | Used by caching plugins. |
| <code>wp</code> | After WP object is set up (ref array) |
| <code>template_redirect</code> | Before determining which template to load. |
| <code>get_header</code> | Before the header template file is loaded. |
| <code>wp_enqueue_scripts</code> | When scripts and styles are enqueued. |
| <code>twentyeleven_enqueue_color_scheme</code> | (Specific to Twenty Eleven) |
| <code>wp_head</code> | Used to print scripts or data in the head tag on the front end. |
| <code>wp_print_styles</code> | Before styles in the \$handles queue are printed. |
| <code>wp_print_scripts</code> | Before scripts in the \$handles queue are printed. |
| <code>get_search_form</code> | |

Code Reference

Browse: [Home](#) / [Reference](#) / Hooks



[\\$action](#)

Action Hook: Fires the requested handler action.

Source: wp-admin/admin-post.php:45

Used by [0 functions](#) | Uses [0 functions](#)

[\\$hook](#)

Action Hook: Fires scheduled events.

Source: wp-cron.php:100

Used by [0 functions](#) | Uses [0 functions](#)

[\\$page_hook](#)

Action Hook: Used to call the registered callback for a plugin screen.

Source: wp-admin/admin.php:211

Used by [0 functions](#) | Uses [0 functions](#)

[\\$per_page](#)

Filter Hook: Filter the number of items per page to show for a specific 'per_page' type.

Source: wp-admin/includes/post.php:971

Used by [0 functions](#) | Uses [0 functions](#)

['handle_bulk_actions-' . get_current_screen\(\)->id](#)

Filter Hook: Fires when a custom bulk action should be handled.

Source: wp-admin/edit-comments.php:98

Used by [0 functions](#) | Uses [0 functions](#)

['handle_network_bulk_actions-' . get_current_screen\(\)->id](#)

Filter Hook: Fires when a custom bulk action should be handled.

Source: wp-admin/network/site-themes.php:140

Used by [0 functions](#) | Uses [0 functions](#)

Koukkuja on PALJON

- WP 2.0: 261 kpl
- WP 3.0: 1259 kpl
- WP 4.0: 1680 kpl
- WP 4.9.2: 2709 kpl

Hyödyllisiä koukkuja

- `pre_option_(asetuksen nimi)`
 - Jokaiselle `get_option()`-funktioilla luettavalle asetukselle on filteri, jonka läpi asetukset menee. Tällä voit muuttaa asetusta lennossa.
- `wp_head`, `wp_footer`
- `the_content`, `the_excerpt`, `the_title`

Hyödyllisiä koukkuja

- `post_class`, `body_class`
 - Lisää CSS-luokkia artikkelin elementille tai dokumentin `<body>`-tagille.
- `pre_get_posts`
 - Tapahtuu ennen kuin sivulla näytettävät artikkelit ladataan. Mahdollistaa esimerkiksi arkistosivulla näytettävien artikkelien säätämisen.

```
add_action('pre_get_posts', 'kirjavinkit_sarjat_jarjesta');
function kirjavinkit_sarjat_jarjesta($query) {
    if (isset($query->query_vars['sarja'])) {
        $query->set("order", "ASC");
    }
}
```

Hyödyllisiä koukkuja

- `wp_enqueue_scripts`
 - Oikea tapa lisätä Javascriptit ja ulkoiset CSS-tyylit sivuille on käyttää tätä koukkuja ja `wp_enqueue_scripts()`-funktiota. Silloin esimerkiksi erilaiset optimointilisäosat pystyvät optimoimaan Javascriptit.

Mitä tästä opittiin?

- Suurta osaa WordPressin toiminnasta on mahdollista muuttaa filttäreillä tai actioneilla.
- Älä koske WordPressiin tai plugineihin, tee muutokset filttäreillä tai actioneilla.
- Jos teet itse pluginia, mieti: "haluaako joku muuttaa tätä arvoa tai toimintoa?" Jos vastaus on edes "ehkä", vedä se filtlerin läpi.
- Jos pluginissa ei ole tarvitsemaasi filtteriä, pyydä kehittäjää lisäämään sellainen!