THEME DEVELOPMENT

Add jQuery Autocomplete to Your Site's Search

by Stephen Harris 15 Apr 2012 32 Comments

The function `get_search_form()` can (and should!) be used to display the search form for your site. It does the job, but it's very bland. Shipped with WordPress since 3.3, however, is a tool which can make using it a lot easier. In this tutorial I'll be showing you how to add jQuery Autocomplete to your search form.

Open Up Your Search Form

This tutorial assumes that your theme uses `get_search_form()` to display your search form and calls `wp_footer()`.

First let's take a look at the TwentyEleven search form (**searchform.php** in your theme). Chances are, yours is very similar. If you can't find **searchform.php** in your theme, then it's probably using the default mark-up which is almost identical. If your search form is hard-coded I'd recommend putting it into **searchform.php**, and using `get_search_form();` to display it.

```
<form method="get" id="searchform" action="<?php echo esc_url( hom
    <label for="s"><?php _e( 'Search', 'twentyeleven' ); ?></labe
    <input type="text" name="s" id="s" placeholder="<?php esc_attr
    <input type="submit" name="submit" id="searchsubmit" value="<?
</form>
```

What we're after is the ID attribute of the search input, so we can target it with jQuery. In this case it's '`s`'.

Before we start, let's do a bit of ground work. This will also serve as a summary of what we shall be doing.

Step 1 Ground Work

All the following should go into your theme's **functions.php**. We are going to hook onto the '`init`' hook with a function that will:

- **Register some CSS and JavaScript** – We're going to need some of the jQuery UI styling. I'm just going to use the basic styling, but you can always roll your own theme to fit in with your site. You can add it to your theme's **style.css** or keep it in a separate file and register it as shown here. We are going to need some custom javascript too, which I will call **myacsearch.js** and store it in my theme's **js** folder.
- **Hook our JavaScript and CSS** – We want to add our styling and javascript when (and only when) the search form is displayed. The `get_search_form` filter fires whenever this happens, and we'll use that to enqueue our scripts and styles.
- **Ajax actions** – We need to add a callback function to process the request and return the results when WordPress receives our action via AJAX. To do this we use the hooks, `wp_ajax_{action}` and `wp_ajax_nopriv_{action}` where `{action}` is used as an identifier for the action we wish to perform (and so should be unique). I shall set it to `myprefix_autocompletesearch`.

```
01 add_action( 'init', 'myprefix_autocomplete_init' );
02 function myprefix_autocomplete_init() {
03     // Register our jQuery UI style and our custom javascript fil
04     wp_register_style('myprefix-jquery-ui','<span class="skimlink
05     wp_register_script( 'my_acsearch', get_template_directory_uri
06 
07     // Function to fire whenever search form is displayed
08     add_action( 'get_search_form', 'myprefix_autocomplete_search_
09 
10     // Functions to deal with the AJAX request - one for logged i
11     add_action( 'wp_ajax_myprefix_autocompletesearch', 'myprefix_
12     add_action( 'wp_ajax_nopriv_myprefix_autocompletesearch', 'my
13 }
```

Step 2 The AJAX URL

We'll be using AJAX to send the search form's input and return the matching posts as the user types. So we will need to give Autocomplete the URL to send the request to. WordPress has a specific URL which deals with AJAX requests, and this is given by `admin_url( 'admin-ajax.php' )`. This gives us the URL on the server side – but we want it in our javascript file. This can be done using `wp_localize_script`. This function was originally intended to help with localisation, but we can repurpose it for our use. Put this immediately after registering the custom javascript `my_acsearch` in step 1:

```
1 wp_localize_script( 'my_acsearch', 'MyAcSearch', array('url' => ac
```

This defines an object `MyAcSearch` which has a property '`url`'. Such a method allows you to send settings stored in the database to the javascript file, but for our

purposes we only need `MyAcSearch.url` which is the URL to direct our AJAX request.

Step 3 The JavaScript

jQuery's autocomplete comes with a fair bit of functionality packed into it, but we'll be sticking to the basics. You can see all its features on the demo page. The data we send to the AJAX URL will include an action variable whose value is the action identifier we set in step 1. In our case it's `myprefix_autocompletesearch`. So, now in our javascript file, add the following.

```
var acs_action = 'myprefix_autocompletesearch';
```

This allows us to identify the request, perform the search and return the results. Next we apply the Autocomplete function to the search form (here we use the ID attribute of the form input):

```
$("#s").autocomplete({
    source: function(req, response){
        $.getJSON(<span class="skimlinks-unlinked">MyAcSearch.url+
    },
    select: function(event, ui) {
        <span class="skimlinks-unlinked">window.location.href=ui.i
    },
    minLength: 3,
});
```

The source function should return an array of objects. Each object should have the property '`label`' (to display in the suggestion list), and we'll be adding the property '`link`', the URL of the post. The select function is fired when a user clicks one of the suggestions. In this example, clicking the suggestion takes you to that page. The `minLength` indicates the number of characters the user must type before the autocomplete kicks in.

We'll wrap this all in a `.ready` handler, so it's only run when the page has fully loaded. Then the complete javascript is:

```
jQuery(document).ready(function ($){
    var acs_action = 'myprefix_autocompletesearch';
    $("#s").autocomplete({
        source: function(req, response){
            $.getJSON(<span class="skimlinks-unlinked">MyAcSearch
        },
        select: function(event, ui) {
            <span class="skimlinks-unlinked">window.location.href
        },
        minLength: 3,
    });
});
```

Step 4 Enqueuing Our Scripts and Styles

Whenever the search form is displayed using the `get_search_form();` function, our function `myprefix_autocomplete_search_form` will fire. In this function we enqueue the scripts and styles that we need for Autocomplete. We don't need to load jQuery or Autocomplete directly, WordPress already knows that we need it and will handle that for us.

```
function myprefix_autocomplete_search_form(){
    wp_enqueue_script( 'my_acsearch' );
    wp_enqueue_style( 'myprefix-jquery-ui' );
}
```

All that remains is to handle the AJAX request.

Step 5 Handling the AJAX Request

Recall that in our `myprefix_autocomplete_init` function we called something like the following:

```
add_action( 'wp_ajax_{action}', 'my_hooked_function' );
add_action( 'wp_ajax_nopriv_{action}', 'my_hooked_function' );
```

The first action is fired when WordPress receives an AJAX request with action given by `{action}` and the user is logged in. The second is fired when the user is not logged in. This can be particularly useful if you only want to process an AJAX request if the user is logged in. For our purposes we want it to work for both logged in and non-logged in users, so we hook our function onto both. Here we define that callback function, again this goes in your **functions.php**:

```
function myprefix_autocomplete_suggestions(){
    // Query for suggestions
    $posts = get_posts( array(
        's' =>$_REQUEST['term'],
    ) );

    // Initialise suggestions array
    $suggestions=array();

    global $post;
    foreach ($posts as $post): setup_postdata($post);
        // Initialise suggestion array
        $suggestion = array();
        $suggestion['label'] = esc_html($post->post_title);
        $suggestion['link'] = get_permalink();

        // Add suggestion to suggestions array
        $suggestions[]= $suggestion;
    endforeach;

    // JSON encode and echo
    $response = $_GET["callback"] . "(" . json_encode($suggestion
    echo $response;

```

```
    // Don't forget to exit!
    exit;
}
```

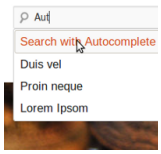
Let's go through this a bit at a time. The input the user has typed is sent along with the AJAX request, and is given by `$_REQUEST['term']`. We simply use `get_posts`' search attribute to search our posts with that term.

We then go through each of the returned posts, and for each one we construct a suggestion array. That array contains the post's title (we call it '`label`' so Autocomplete will recognise it and use it for the suggestion list) and the post's permalink, so that clicking a post will direct the user to that page. We add each suggestion array to a suggestions array.

Next, we JSON encode the suggestions, and echo the result. Finally, we exit! And we're done!

Let us know what you think in the comments, and if you have any suggestions on how to extend on this, we'd love you to share those too.

Difficulty:
Beginner
Length:
Quick
Categories:



Theme Development | WordPress | jQuery

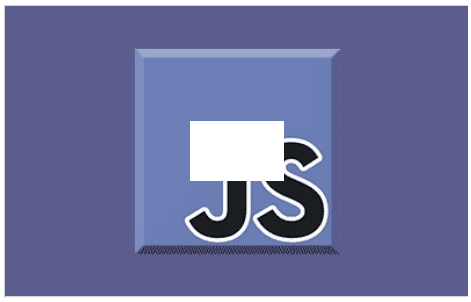
Translations Available:

Tuts+ tutorials are translated by our community members. If you'd like to translate this post into another language, let us know!

About Stephen Harris

Stephen is a Mathematician, Christian & WordPress developer all rolled into one, oddly human shape. He's the author of the event management plug-in Event Organiser.

Suggested Tuts+ Course



JavaScript for PHP Developers $15

Related Tutorials

Fifty Actions of WordPress – 50 Examples (31 to 40)
Code

Token-Based Authentication With AngularJS & NodeJS
Code

Building With the Twitter API: OAuth, Reading and Posting
Code

Jobs

Web Developer / Site Management
at Zanes Law in Phoenix, AZ, USA

PHP Coder with Magento Knowledge
at Yoginet Web Solutions in New Delhi, Delhi, India

Envato Market Item

32 Comments Wptuts+ Login

Sort by Best Share Favorite

Join the discussion…

Patil Makarand • 2 years ago
Nice tutorial , but I am looking search box for Wordpress ! :(
But got overview of its working. Thanks
1 • Reply • Share ›

Louis_Dea • 3 years ago
Hi, Thanks for posting this script!

On my side, I can't make the autocomplete script worked. When I'm writing in the form, I get this message in FireBug : " MyAcSearch is not defined : [Break On This Error] : $.getJSON(MyAcSearch.url+'?callback=?&action='+acs_action, req, response)".

Any idea on what I'm doing wrong?

Thank you for your time!
1 • Reply • Share ›

Louis_Dea ➚ Louis_Dea • 3 years ago
Sorry, my bad. I forgot to include one line of the tutorial...
• Reply • Share ›

Sarasota Website Design ➚ Louis_Dea • 3 years ago
I am having the same problem with MyAcSearch not defined in the js. Where does this get added?

Thanks
• Reply • Share ›

Stephen Harris ➚ Sarasota Website Design • 3 years ago
This gets added with the `wp_localize_script` function in Step 2. This should be placed immediately after the `wp_register_script` in Step 1.
• Reply • Share ›

Soufiane • 3 years ago
Why there's no working wp DEMO ????
1 • Reply • Share ›

Mahesh Waghmare • 7 months ago
Nice Tutorial...!
It Works.... Just wait for processing ajax request.

@Stephen Harris please add loading gif to show request is in process. But Awesome work... Thanks....
• Reply • Share ›

Loren • a year ago
Thanks for the tutorial. This is not working for me when logged out. Anyone else have this issue?
• Reply • Share ›

Faisal Abu Jabal • a year ago
Man you are awesome !

• Reply • Share ›

Lucas Bustamante • a year ago
I want to show pages instead of posts

Tried this but didn't work
http://pastebin.com/urP5xbz6
• Reply • Share ›

Mohamed • 2 years ago
Thanks for the awesome tutorial however wordpress default search is horrible! i'd like do an Ajax Get request for the /?s=keyword&lang=en with my keyword and parse the results these way you can get the most out of your autocomplete search.

Another reason why i recommend this way is because wordpress don't search inside custom fields by default.

Thanks,
Mohamed
• Reply • Share ›

BretJG • 3 years ago
Sweet tutorial - I like the auto complete on Magento's search, definitely going to add it to my wordpress sites from here on out.
• Reply • Share ›

Sarasota Website Design • 3 years ago
How would you set it up to also search page titles and not just just post types?
• Reply • Share ›

Stephen Harris ➚ Sarasota Website Design • 3 years ago
'Page' is just a post type. However, WordPress searches content, not just titles. If you want to search only titles, then you might want to read my response to Brad.
• Reply • Share ›

Kyle • 3 years ago
Great stuff - got it working just fine. One question...is there an easy way to add custom post types to the results?
• Reply • Share ›

Kyle ➚ Kyle • 3 years ago
Got this figured out by adding a filter to the default search to include custom post types....

```
function filter_search($query) {
if ($query->is_search) {
$query->set('post_type', array('post', 'events'));
};
return $query;
};
add_filter('pre_get_posts', 'filter_search');
```

1 • Reply • Share ›

Vijay • 3 years ago
Very Nice tutorial. I am building a website and will try it when I get home tonight.
• Reply • Share ›

Paul • 3 years ago
Nice tutorial it's good idea think I'll give it a go on mine.

How many posts in your blog have you tested this with?
• Reply • Share ›

Rajendra Banker • 3 years ago
Very Use full Tutorial!

Brad Brevet • 3 years ago

How would you go about altering the code to get an exact search for the term being input? I am setting this up to query movie titles so it would really help to limit any vague matches.

Reply • Share ›

Stephen Harris ➚ Brad Brevet • 3 years ago

WordPress doesn't offer this functionality by default. This answer shows you how to do that. That will, though, effect every search on your site - if you only want it to effect the AJAX searches (i.e. the jQuery Autocomplete searches), then at the top of that function I've linked to you'll want to check the 'DOING_AJAX' constant eg:

```
if (defined('DOING_AJAX') && DOING_AJAX)
return $search;

//continue with the rest of the function
```

Hope that helps :)

Reply • Share ›

Maor Chasen • 3 years ago

A great tutorial, but ... oops! no input validation.

```
$posts = get_posts( array(
's' =>$_REQUEST['term'],
) );
```

really?!

Reply • Share ›

Stephen Harris ➚ Maor Chasen • 3 years ago

Yup, the received input is not outputted anywhere so no sanitisation is required and WordPress handles all the escaping for the database query - so you can pass it straight to get_posts.

Reply • Share ›

Kriesi • 3 years ago

Nice Tutorial and definitely a good start for an autocomplete script but as Nicolas said: querying the database with each keystroke may kill an already busy server quite fast.

Would love to see some suggestions on how to cache the first query and then check this "cached version" instead of the database. What comes to my mind would be to query only after the 2nd or 3rd keystroke and if you get a result thats not to big for javascript to handle (keep in mind there are blogs and magazines with thousands of posts) filter that result instead of querying the database with every new keystroke...

Reply • Share ›

arifur rahman ➚ Kriesi • 3 years ago

Kriesi
There is a jquery plugin by using that you can control ajax fire. Like untile a min key word is type and the user stop typing. When the user stop typing and exit min key word limit then the funciton will fire.

Reply • Share ›

Japh ➚ Kriesi • 3 years ago

Hey Kriesi, looking in the JavaScript snippets for Step 3, I see that jQuery Autocomplete will only query the database with at least 3 characters entered, which is a bit better and will return a smaller result set. However, I agree that it would be great to see a solution with caching and filtering as mechanisms for reducing the database load this kind of feature can introduce.

Thanks for the feedback! :)

Reply • Share ›

Anthony • 3 years ago

Thanks!

Reply • Share ›

Kazumi ➚ Anthony • 3 years ago

Im typing exlcay what he types in the first 5 mins, but when i load the browser, the tab title is chat , like its suposed to be, but there is no body . Its all blank. Someone know why?

• Reply • Share ›

Duane • 3 years ago

Brilliant tutorial, been looking for a way to jazz up the old search system, this might just be it :-D thanks!

• Reply • Share ›

Nicolas • 3 years ago

Nice tutorial. Thanks. Easy and efficient.

Main drawback for this approach is that when you get to have quite some readers, searching into your posts for every keystroke will become heavy. Index search will become more appropriate. But we talk about heavy stuff.

Is this function:
$posts = get_posts(array(
's' =>$_REQUEST['term'],
));

Full text search? Or is it a search on the titles?

• Reply • Share ›

Wpfix ➚ Nicolas • 3 years ago

Nice Tutorial.

• Reply • Share ›

Stephen Harris ➚ Nicolas • 3 years ago

Very good point. This is why I recommend setting 'minLength' to 3 - that way only after three characters are entered is a search performed. This increases the chances of the desired post appearing in the suggestion list first time. Of course, subsequent letters will require another search - but if the post has already appeared then the user will probably select it.

The search performs the standard WordPress search :).

Thanks for posting!

• Reply • Share ›

Teaching skills to millions worldwide. 18,893 Tutorials 461 Video Courses