Shopify Search Filter Project (still in testing)
By, David Weru

A client had a site with a wide range of content, from articles to products and everything in between.

Whenever a user were to search a term on one of the store's many search elements, the site would return results from articles to products and everything in between.

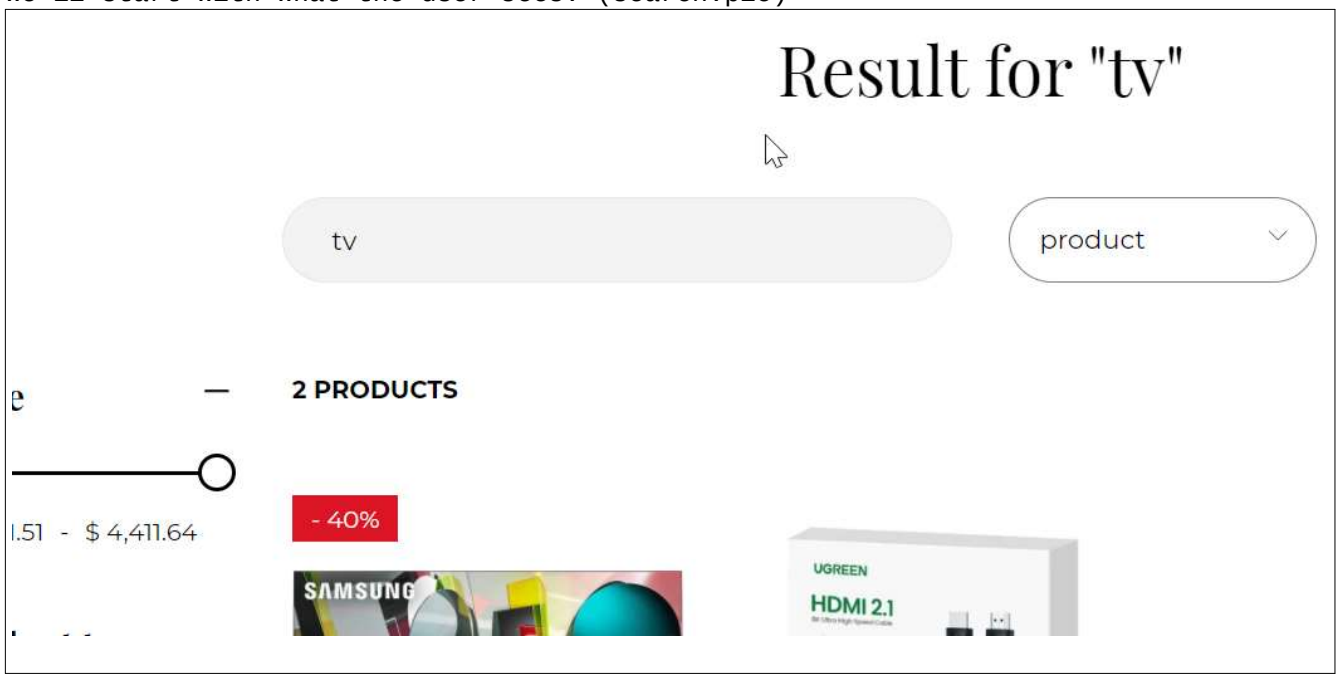
The client scoured the administration panels for settings to filter search results to no avail. Afterwards, the client went under the hood and looked at the code, which of course yielded nothing.

This led the client to reach out for anyone who could help.

This is where our story begins.

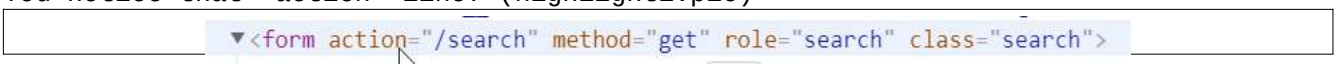
Before going further, lets explain a few technical things about how search works on shopify.

We'll start with what the user sees. (search.pic)



This is a search bar. In the code, it's just a form. It looks like this. (form.pic)

You notice that “action” line? (highlight1.pic)



This is where your searches are sent to.

Now, let’s go back stage and see what kind of things you can send to this page. (parameters.pic)

Resource type	Searchable properties
Products	▪ body ▪ product_type ▪ tag ▪ title ▪ variants.barcode ▪ variants.sku ▪ variants.title ▪ vendor
Pages	▪ author ▪ body ▪ title
Articles	▪ author ▪ body ▪ tag ▪ title
Collections	▪ title

Now we are interested in these two columns above in particular.

The fields needed in the form to filter are named: “type” and “fields”.

“type” corresponds with “resource type” above.
“fields” corresponds with “searchable properties” above.

Within these two parameters, you can send what results you want returned.

These are sent as comma separated strings.

This is what it looks like when you want everything returned (explicitly)

```
<form action="/search" method="get" class="center-block por" role="search" style="max-width:600px;margin-bottom:30px;">
  <input class="search-full__input" type="search" name="q" placeholder="Seek" aria-label="Seek" data-term>
  <input type="hidden" name="options[prefix]" value="last" aria-hidden="true">
  <button type="submit" class="search-full__submit">…</button>
  <input name="type" type="hidden" value="product,page,article">
  <input name="fields" type="hidden" value="title,author,body,product_type,tag,variants.barcode,variants.sku,variants.titl
  e,vendor"> == $0
</form>
```

(chain1.pic)
(chain2.pic)

If you don't make specifications, everything gets returned also, which is where we started. Without a filter.

These are various filtration examples

(various filtration examples.pic)

```
<input name="type" type="hidden" value="page"> == $0
<input name="fields" type="hidden" value="title,author">
```

```
<input name="type" type="hidden" value="product,article"> == $0
<input name="fields" type="hidden" value="title,author,variants.barcode,variants.sku,variants.title,vendor">
```

Ok great! So there's a second part to this problem.

There are many search forms on this store, and not all of those forms need this filter.

These forms have class identifiers.

```
<form action="/search" method="get" class="center-block por" role="search" style="max-width:600px;margin-bottom:30px;">
```

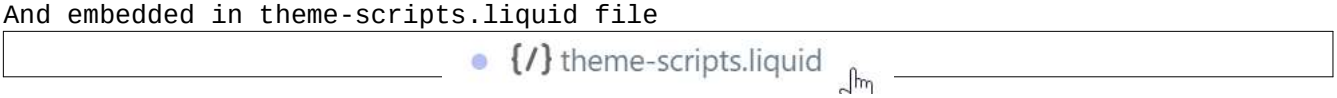
This one has "center-block" and "por". You can add your own, if you like.

If you know which form you wish to apply the filter to, you can effectively have the filter added in for you via javascript.

Below is a simple script I wrote which runs on all pages as it lives in the header.

The script is stored in the assets folder,

And embedded in theme-scripts.liquid file




```
11  <script src="{{ 'search_filter.js' | asset_url }}" defer="defer"></script>
```

Below is the short script with a quick description of what it does.

search_filter.js

Delete file | Rename | Sav

```
function search_filter_v1(form_class_name,type_filter,fields_filter){
   var xlx = document.getElementsByTagName("form");
  for(a=0;a<xlx.length;a++){if(xlx[a].getAttribute("action")=="/search"){
     if( xlx[a].getAttribute("class").split(' ').includes(form_class_name)){

      s_u1=xlx[a].getElementsByTagName("input");
      for(b=0;b<s_u1.length;b++){
        s_u1[b].getAttribute("name")=="type"?s_u1[b].remove():'';
        s_u1[b].getAttribute("name")=="fields"?s_u1[b].remove():'';}

      var inp = document.createElement("input");
      inp.setAttribute("name","type");
      inp.setAttribute("type","hidden");
      inp.setAttribute("value",type_filter);
      xlx[a].appendChild(inp);

       var inp = document.createElement("input");
      inp.setAttribute("name","fields");
      inp.setAttribute("type","hidden");
      inp.setAttribute("value",fields_filter);
      xlx[a].appendChild(inp);
    }
   }
  }

}
```

What it does:

1: Gather all form elements on the current page.

2: Find only search form elements.

3: Find only the search forms with the class name identifier.

4: Remove any filters already on the form.

5: Create the type filter and add filter string.

6: Create fields filter and add filter string.

7: Add the filters to the form.

This is how it is used:

```
search_filter_v1("search__input","product","title,author,body,product_type,tag,variants.barcode,variants.sku,variants.title,vendor");
```

The format is:

```
search_filterv1(“class_name”,”type_string”,”fields_string”);
```

You simply add this line to the bottom of the javascript file. You can have one for each form which you wish to modify.

It’s written in without the need for dependencies in case some stores do not have

or want jquery or another javascript library installed. It will not slow down page loads.

To recap, now pending testing, the client will be able to finely filter results on a form by form basis simply by writing out one line of code.

For this project, there is still a possibility of adding a user interface in the theme customizer but that depends on weather the functionality pans out in all cases and weather time allows.