**DESCRIPTION:**

Build a simple search page using **HTML5, CSS and Javascript**. What you build is entirely up to you, however, the application should hit the provided search service via AJAX, display the results, and of course meet the requirements. You may spend as much time as you feel comfortable spending within the allotted time constraints.

**REQUIREMENTS:**

Toolbox:

HTML5

CSS3

jQuery *(Feel free to use the Google CDN by including this tag in your markup <script src="//ajax.googleapis.com/ajax/libs/jquery/1.11.0/jquery.min.js"></script>)*

Git *(create an account on github.com, or use existing)*

Frontend:

* Markup must be valid, **semantic** HTML5
* The page must contain a form with text box for the user to search, and a button to submit
* The form must contain simple javascript validation to ensure the text box is not empty and contains at least 3 characters
* The form must submit to the search service asynchronously ( feel free to use jQuery's ajax function for this as it will save you time )
* Errors must be handled and a friendly message displayed to the user should one be encountered
* The page must display the search results in a user-friendly way ( "**Name**", "**SKU**", "**Price**", "**Manufacturer**", and "**Description**" are required )
* The page must Use CSS to style the page. It doesn't need to win any beauty pageants, but shouldn't be a plain white page either. Simple styling is fine here. It’s your search page, do whatever you’d like.
* Our market research has shown that mobile users are more likely to use search forms if they have cats for background images. Use a simple media query to change the page’s background to a funny cat image if the user is on a mobile device.

Bonus: *(not required, only implement if you have the time and/or are bored):*

* Allow for sorting the results. Valid sort options and values are below: [refers to API parameter (sort\_by\_field)]

*Option* *Value*

Name Ascending Name:ASC

Name Descending Name:DESC

Price Ascending Price:ASC

Price Descending Price:DESC

Relevance (default, no value)

* The service returns 25 results per call. If the total results > 25, please create a simple pagination system to show the next page(s) of results

Search API:

*URL*: http://api.americanmuscle.com/search/get

*Method*: GET

*Parameters*:

(string) keywords (required, minimum length 3)

(int) page (optional, defaults to 1 if not passed)

(string) sort\_by\_field (optional, ENUM( 'Price:ASC', 'Price:DESC', 'Name:ASC', 'Name:DESC' ), defaults to empty if not passed )

(int) json (optional, defaults to 0 if not passed)

Sample request: http://api.americanmuscle.com/search/get?keywords=catback+exhaust&sort\_by\_field=Price:ASC&json=1

[Sample keywords: catback+exhaust,tuner,keychain (pretty much anything on our website will work here)]

*Response*: JSON (application/json) [if json=0, then XML text/xml]

*Error Codes*:

* 400 occurs due to a bad request (invalid parameters, etc)
* 500 occurs due to a fatal server error

The response returns a bunch of stuff that you will not use. The important stuff is outlined below:

*Total results*: json.pagination.total\_products

*Items per page*: 25 (this is hardcoded, so you'll always get <= 25 results )

*Current page*: json.pagination.current\_page (page parameter if passed)

*Total pages*: json.pagination.total\_pages

*Results*: json.results (array)

*Name*: json.results[n].Name

*SKU*: json.results[n].Sku

*Price*: json.results[n].Price

*Manufacturer*: json.results[n].Manufacturer

*Description*: json.results[n].Description