jQuery Typeahead Search: Documentation

VERSION 2.10.4 3 months ago

Learn how to use the JavaScript Typeahead Search wrapped in a jQuery plugin.

Issue or Request

Sambungan Aman Gagal

DOWNLOAD

Installation

Use Bower package manager to get Typeahead

```
# Get the latest version
$ bower install jquery-typeahead
```

OR

Use NPM package manager to get Typeahead

```
# Get the latest version
$ npm install jquery-typeahead
```

Required

- jQuery 1.7.2 and above because of `.on` event handler - Download
- Latest Typeahead plugin 2.10.4 - Download

```
<html>
<head>
    ...
    <!-- Optional CSS -->
    <link rel="stylesheet" href="/vendor/jquery-typeahead/dist/jquery.typeahead.min.css">

    <!-- Required JavaScript -->
    <script src="https://code.jquery.com/jquery-2.1.3.min.js"></script>
    <script src="/vendor/jquery-typeahead/dist/jquery.typeahead.min.js"></script>
    ...
</head>
```

Initialization

There are 2 ways to initialize the typeahead search:

- Call the `$.typeahead()` function with the `input` option configured [recommended]

```
$.typeahead({
    input: ".js-typeahead",
    order: "asc",
    source: {
        groupName: {
            // Ajax Request
            ajax: {
                url: "..."
            }
        }
    },
    callback: {
        onClickBefore: function () { ... }
    }
});
```

OR

- Create a jQuery object using a **unique input** selector. Then chain the `.typeahead()` function containing parameters as an object.

```
$('.js-typeahead').typeahead({
    order: "asc",
    source: {
        groupName: {
            // Array of Objects / Strings
            data: [ {...}, {...} ]
        }
    },
    callback: {
        onInit: function () { ... }
    }
});
```

HTML Structure and CSS

The Typeahead plugin requires a specific HTML structure. This offers some advantages:

- Easy HTML integration
- Default styling using `/dist/jquery.typeahead.min.css`
- Bootstrap 3 and 4 ready
- * As of 2.5.0 Typeahead is using the BEM css standards

```
<form>
    <div class="typeahead__container">
        <div class="typeahead__field">

            <span class="typeahead__query">
                <input class="js-typeahead"
                       name="q"
                       type="search"
                       autocomplete="off">
            </span>
            <span class="typeahead__button">
                <button type="submit">
                    <span class="typeahead__search-icon"></span>
                </button>
            </span>

        </div>
    </div>
</form>
```

Configuration

The user's configuration object will be merged with the default plugin configuration.

```
/**
 * @private
 * Default options
 */
var _options = {
    input: null,              // *RECOMMENDED*, jQuery selector to reach Typeahead's input for initializat
    minLength: 2,             // Accepts 0 to search on focus, minimum character length to perform a searc
    maxLength: false,         // False as "Infinity" will not put character length restriction for searchi
    maxItem: 8,               // Accepts 0 / false as "Infinity" meaning all the results will be displayed
    dynamic: false,           // When true, Typeahead will get a new dataset from the source option on eve
    delay: 300,               // delay in ms when dynamic option is set to true
    order: null,              // "asc" or "desc" to sort results
    offset: false,            // Set to true to match items starting from their first character
    hint: false,              // Added support for excessive "space" characters
    accent: false,            // Will allow to type accent and give letter equivalent results, also can de
    highlight: true,          // Added "any" to highlight any word in the template, by default true will c
    multiselect: null,        // Multiselect configuration object, see documentation for all options
    group: false,             // Improved feature, Boolean,string,object(key, template (string, function))
    groupOrder: null,         // New feature, order groups "asc", "desc", Array, Function
    maxItemPerGroup: null,    // Maximum number of result per Group
    dropdownFilter: false,    // Take group options string and create a dropdown filter
    dynamicFilter: null,      // Filter the typeahead results based on dynamic value, Ex: Players based or
    backdrop: false,          // Add a backdrop behind Typeahead results
    backdropOnFocus: false,   // Display the backdrop option as the Typeahead input is :focused
    cache: false,             // Improved option, true OR 'localStorage' OR 'sessionStorage'
    ttl: 3600000,             // Cache time to live in ms
    compression: false,       // Requires LZString library
    searchOnFocus: false,     // Display search results on input focus
    blurOnTab: true,          // Blur Typeahead when Tab key is pressed, if false Tab will go though searc
    resultContainer: null,    // List the results inside any container string or jQuery object
    generateOnLoad: null,     // Forces the source to be generated on page load even if the input is not f
    mustSelectItem: false,    // The submit function only gets called if an item is selected
    href: null,               // String or Function to format the url for right-click & open in new tab or
    display: ["display"],     // Allows search in multiple item keys ["display1", "display2"]
    template: null,           // Display template of each of the result list
    templateValue: null,      // Set the input value template when an item is clicked
    groupTemplate: null,      // Set a custom template for the groups
    correlativeTemplate: false, // Compile display keys, enables multiple key search from the template strir
    emptyTemplate: false,     // Display an empty template if no result
    cancelButton: true,       // If text is detected in the input, a cancel button will be available to re
    loadingAnimation: true,   // Display a loading animation when typeahead is doing request / searching f
    filter: true,             // Set to false or function to bypass Typeahead filtering. WARNING: accent,
    matcher: null,            // Add an extra filtering function after the typeahead functions
    source: null,             // Source of data for Typeahead to filter
    callback: {
        onInit: null,             // When Typeahead is first initialized (happens only once)
        onReady: null,            // When the Typeahead initial preparation is completed
        onShowLayout: null,       // Called when the layout is shown
        onHideLayout: null,       // Called when the layout is hidden
        onSearch: null,           // When data is being fetched & analyzed to give search results
        onResult: null,           // When the result container is displayed
        onLayoutBuiltBefore: null,  // When the result HTML is build, modify it before it get showed
        onLayoutBuiltAfter: null,   // Modify the dom right after the results gets inserted in the result co
        onNavigateBefore: null,     // When a key is pressed to navigate the results, before the navigation
        onNavigateAfter: null,      // When a key is pressed to navigate the results
        onEnter: null,              // When an item in the result list is focused
        onLeave: null,              // When an item in the result list is blurred
        onClickBefore: null,        // Possibility to e.preventDefault() to prevent the Typeahead behaviors
        onClickAfter: null,         // Happens after the default clicked behaviors has been executed
        onDropdownFilter: null,     // When the dropdownFilter is changed, trigger this callback
        onSendRequest: null,        // Gets called when the Ajax request(s) are sent
        onReceiveRequest: null,     // Gets called when the Ajax request(s) are all received
        onPopulateSource: null,     // Perform operation on the source data before it gets in Typeahead data
        onCacheSave: null,          // Perform operation on the source data before it gets in Typeahead cach
        onSubmit: null,             // When Typeahead form is submitted
        onCancel: null              // Triggered if the typeahead had text inside and is cleared
    },
    selector: {
        container: "typeahead__container",
        result: "typeahead__result",
        list: "typeahead__list",
        group: "typeahead__group",
        item: "typeahead__item",
        empty: "typeahead__empty",
        display: "typeahead__display",
        query: "typeahead__query",
        filter: "typeahead__filter",
        filterButton: "typeahead__filter-button",
        dropdown: "typeahead__dropdown",
        dropdownItem: "typeahead__dropdown-item",
        labelContainer: "typeahead__label-container",
        label: "typeahead__label",
        button: "typeahead__button",
        backdrop: "typeahead__backdrop",
        hint: "typeahead__hint",
        cancelButton: "typeahead__cancel-button"
    },
    debug: false // Display debug information (RECOMMENDED for dev environment)
};
```

Option	Type	Description
input	{string} [optional]	The jQuery input selector is only required if the Typeahead was initialized without a jQuery object. In that case, if no ir Typeahead is dropped.
minLength	{numeric}	**2** (default) The number of characters typed inside the search input before searching for results. It is possible to set this option to `searchOnFocus: true` to display a set of results by default.
maxLength	{numeric}	**false** (default) The maximum number of characters typed inside the input to perform a search.

maxItem	{numeric}	**8** (default)
		The maximum number of search results that will appear inside the list. Set 0 to display ALL search results.
		It is possible to combine `maxItem` with `maxItemPerGroup` to get different results.
		`// Only 8 results will be displayed`
		`// Could be 6 from group1 and 2 from group2 (8 results)`
		`maxItem: 8,`
		`source: {`
		`  group1: {},`
		`  group2: {},`
		`  group3: {},`
		`  group4: {}`
		`}`
		
		`// Display "Infinity" number of results until 8 item per group is reached`
		`// Could be 8 from group1, 4 from group2, 1 from group3 and 8 from group4 (21 results)`
		`maxItem: 0,`
		`maxItemPerGroup: 8,`
		`source: {`
		`  group1: {},`
		`  group2: {},`
		`  group3: {},`
		`  group4: {}`
		`}`

dynamic Advanced	{boolean}	**false** (default) By default, the typeahead will only load once the `source` data. Although you can change this behavior and request th every "input" event (similar to keypress). - Demo * Note that you can modify the Ajax request to send the query with `{{query}}` modifier.
delay	{numeric}	**300** (default) If `dynamic: true`, the delay (in milliseconds) will add a timer to prevent from sending a request on every key typed. In be sent once the delay expires.
order	{null|string}	**null** (default) Takes the default order in which the data was given. If "asc" or "desc" is set, they results will re-order based on `displ` **asc** Reorder the result set ascending

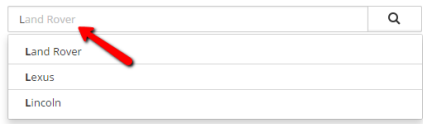
desc
Reorder the result set descending

offset {boolean}

false (default)
The position of the matched query can be anywhere inside the result string
true
The query can only be match if the result string starts by the query characters.

hint {boolean|object}

false (default)
No hint is activated
true or css object
A suggestion text will appear if there is an item inside the result list that starts by the user query, also pressing the rig
search input text will autocomplete the query with the suggested hint and call `callback.onClickBefore` with the selec



accent
Advanced

{boolean|object}

false (default)
If enabled, the query and search results will ignore accents (ãàáâäẽèéêëìíîïõòóôöùúûüñç) and display every matches r
punctuation. Ex: é = e, Ä = a, etc.
It is possible to set a custom accent object, by simply setting `from` and `to` keys
* Using this option on large data sets (10,000+ items) might impact your search performances.

```
// {{group}}" will be replaced by the value of "conference"
accent: {
    from: 'àé',
    to: 'ae'
}
```

highlight {boolean|string}

true (default)
The search result(s) will receive the `<strong>` HTML tag around the matched query.
If set to `true`, only the display keys will be highlighted
If set to `"any"`, any string within the `template` will be highlighted

multiselect {object}

null (default)
`multiselect` configuration:

- **limit** {number}: Optional limit of maximum items to select
- **limitTemplate** {string|function}: Template when the limit is reached
- **matchOn** {string|array}: Unique item identifier to remove an item from the result list when selected (use any of th
JSON of the item will be used
- **cancelOnBackspace** {boolean}: If true, the last label will be deleted if the query is empty and backspace is presse
- **href** {string|function}: Href link on the multiselect item
- **data** {array|function}: Default items when Typeahead is loaded
- **callback.onClick(node, item, event)** {function}: Triggered when a multiselect item is clicked
- **callback.onCancel(node, item, event)** {function}: Triggered when a multiselect item is canceled

```
multiselect: {
    limit: 2,
    limitTemplate: 'You can\'t select more than 2 teams',
    matchOn: ["id"],
    data: function () {
        var deferred = $.Deferred();

        // Load your custom data and resolve the promise
        // setTimeout is only added to simulate a delay
        // see hockey_v2 demo for a working example
        setTimeout(function () {
            deferred.resolve([{
                "matchedKey": "name",
                "name": "Canadiens",
                "img": "canadiens",
                "city": "Montreal",
                "id": "MTL",
                "conference": "Eastern",
                "division": "Northeast",
                "group": "teams"
            }]);
        }, 2000);

        deferred.always(function () {
            console.log('data loaded from promise');
        });

        return deferred;
    },
    callback: {
        onClick: function (node, item, event) {
            event.preventDefault();
            console.log(item);
            alert(item.name + ' Clicked!');
        },
        onCancel: function (node, item, event) {
            console.log(item)
        }
    }
},
```

group
Option changed in 2.5.0

{boolean| string| object}

false (default)
If set to `true`, the results will grouped by the group name specified inside `source`.
If set to `string`, the results will be grouped by the corresponding object key.
Ex: `group: "conference"` will group the hockey teams by "Western" and "Eastern" conferences in hockey_v1 demo
If an Object is set:
- key: Grouping key
- template: Grouping template in the result list (custom header text), can be string or function.

```
// Will use the options.source.groups
group: true

// Will use the object key to group the items, Western or Eastern conferences
group: "conference"

// {{conference}} will be replaced by the value of "conference"
group: {
    key: "conference",
    template: "{{conference}} conference Hockey Teams"
}

// Divisions translated to French
// Function to enable custom group headers based on item properties
// IMPORTANT: Return a string
group: {
    key: "division",
    template: function (item) {
        var groups = {
            "Southeast": "Sud-Est",
            "Southwest": "Sud-Ouest",
            "Northeast": "Nord-Est",
            "Northwest": "Nord-Ouest",
            "Central": "Centrale",
            "Pacific": "Pacifique",
            "Atlantic": "Atlantique"
        };
        return "Division " + groups[item.division];
```

```
    }
}

// An alternative could be to have a different template if "conference == Eastern"
group: {
    key: "conference",
    template: function (item) {

        var conference = item.conference;
        if (conference.toLowerCase() === "eastern") {
            conference += " Yay!";
        }

        return conference;
    }
}
```

groupOrder — {string| array| function}

null (default)

By default, the groups will be output in the same order as they are defined in `source`

Set "asc" or "desc" to have the group name sorted ascending or descending

Set an Array to specify the group order to appear in the search result

```
groupOrder: ["country","state","continent","capital"],
source: {
    continent: {},
    country: {},
    capital: {},
    state: {}
}
```

Set a Function that returns an Array

```
// In this example, the group will be order by the MOST results first.
// Ex: If there are 5 country matches with "a" and 2 matches with continent,
// country group will appear first in the search result
// Using "this.", see the EXTRAS section for more details
groupOrder: function (node, query, result, resultCount, resultCountPerGroup) {

    var scope = this,
        sortGroup = [];

    for (var i in result) {
        sortGroup.push({
            group: i,
            length: result[i].length
        });
    }

    sortGroup.sort(
        scope.helper.sort(
            ["length"],
            false, // false = desc, the most results on top
            function (a) {
                return a.toString().toUpperCase()
            }
        )
    );

    return $.map(sortGroup, function (val, i) {
        return val.group
    });
}
```

maxItemPerGroup — {number}

false (default)

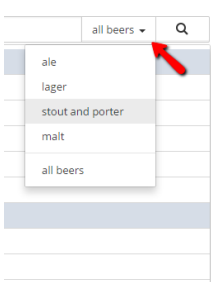
Set a maximum results per group if `group: true` configuration is enabled. - Demo

dropdownFilter — {boolean| string| object| array}

Option changed in 2.5.0

false (default)

If a string a specified, a dropdown filter will be created between the search input and the search submit button using indexes. The string value will appear at the end of the dropdown and will filter through all the sources.



JavaScript | HTML | PHP

```
$('#beer_v1-query').typeahead({
    minLength: 1,
    maxItem: 15,
    order: "asc",
    hint: true,
    group: [true, "{{group}} beers
    maxItemPerGroup: 5,
    backdrop: {
        "background-color": "#fff"
    },
    href: "/beers/{{group}}/{{dis
    dropdownFilter: "all beers",
    emptyTemplate: "No result for
    source: {
        ale: ["/jquerytypeahead/b
        lager: ["/jquerytypeahead/
        "stout and porter": ["/jq
        malt: ["/jquerytypeahead/
    },
```

If an array of objects is set, the dropdownFilter will be built based on it. It is then possible to create filters based on ite

Demo

```
// Will use the source groups as the filters
dropdownFilter: true

// Will use the source groups as the filters but overrides the default text
dropdownFilter: "Show All"

// Only works with "static data" (options.dynamic = false)
// Typeahead will gather every "conference" possibilities on the source items and build the options
// Notice the "value" key is not defined
dropdownFilter: {
    key: 'conference',
    template: '<strong>{{conference}}</strong> Conference Teams',
    all: 'All Conferences'
}

// For dynamic data (options.dynamic = true), the key + value must be specified
// Use an Array to define the filters
// "all" option has to be declared once to override the default text to display all groups
dropdownFilter: [{
    key: 'conference',
    value: 'Western',
    template: '<strong>{{conference}}</strong> Conference Teams'
}, {
    key: 'conference',
    value: 'Eastern',
    template: '<strong>{{conference}}</strong> Conference Teams',
    all: 'All Conferences'
}]

// For "static data" (options.dynamic = false), it is possible to have both ways
// of defining filters in an array
// Notice the missing "value" key inside the first object
dropdownFilter: [{
    key: 'conference',
    template: '<strong>{{conference}}</strong> Conference Teams',
    all: 'All Conferences'
}, {
    key: 'division',
    value: 'Northeast',
    template: '<strong>{{division}}</strong> Division'
}]
```

dynamicFilter — {array} of objects

Advanced

false (default)

If filters objects are defined, the Typeahead source data will be filtered based on the "selected" / "checked" checkboxe based on "OR" and "AND" filtering similar to database queries.

- `selector` is the jquery selector to reach the unique input type "checkbox", "radio" or select tag
- `key` the object key that will be filtered, you can use "dot" notation to reach deep object properties, but in that cas performed. Ex `object.key.is.deep`

- `|` key prefix means "OR" filtering, the object key **CAN** match the value
- `&` key prefix means "AND" filtering, the object key **HAS** to match the value

See game_v3 demo for a complete example

```
<input type="checkbox" id="assassin">
```

```
// If the checkbox is "checked", champions with Assassin tag will be searchable
dynamicFilter: [{
    selector: "#assassin",
    key: "|tags.Assassin"
}],
source: {
    champion: {
        data: [{
            id: 84,
            key: "Akali",
            tags: {
                Assassin: true
            }

        }, {
            id: 222,
            key: "Jinx",
            tags: {
                Marksman: true
            }
        },
        ...
        ]
    }
}
```

backdrop {boolean|object}

false (default)
If set to `true`, html will be added to add a backdrop under the search input and results. It is possible to override the object to this option.

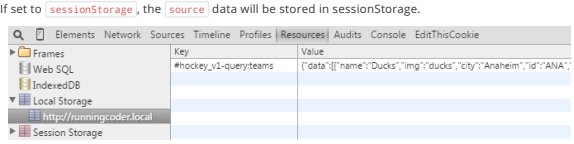
```
// Custom light blue backdrop
backdrop: {
    "opacity": 0.15,
    "filter": "alpha(opacity=15)",
    "background-color": "#eaf3ff"
}
```

backdropOnFocus {boolean}

false (default)
If set to `true`, as soon as the Typeahead input is focused, the `backdrop` option will be displayed regardless.

cache [recommended] {boolean|string}

false (default)
If set to `true` or `localStorage`, the `source` data will be stored in localStorage.
If set to `sessionStorage`, the `source` data will be stored in sessionStorage.



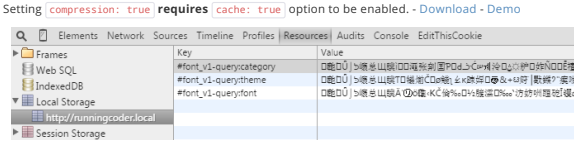
* This option can't be combined with `dynamic: true`

ttl {numeric}

3600000 (1 hour) (default)
* This is a `cache` configuration, it sets the storage time to live in milliseconds.

compression Advanced {boolean}

false (default)
Enabling this option will compress the data inside Localstorage.
Setting `compression: true` **requires** `cache: true` option to be enabled. - Download - Demo



* This option can't be combined with `dynamic: true`

suggestion {boolean|object}

false (default)
* Not yet implemented *

searchOnFocus {boolean}

false (default)
If enabled, the typeahead will display results (if any) on input focus. You can combine this option with the input attribu results when the page is loaded.

blurOnTab {boolean}

true (default)
Blur Typeahead when Tab key is pressed, if false Tab will go though search results

resultContainer {jquery selector|false}

null (default)
If a jQuery string selector or jQuery object is specified, the typeahead results will appear in that container instead of th If the option is set to `false`, the HTML result list will not be built. Use this option for filtering "already on page" HTML hide.

generateOnLoad {boolean}

null (default)
If enabled, the source data will be generated (doing Ajax request and preparing to data to be searched) on page load input to be focused.
* This option does not work well with `dynamic: true` unless some other configuration is adjusted.

mustSelectItem {boolean}

false (default)
If enabled, an item will HAVE TO be selected in order to submit the form. Use this option if you want to restrict search

href {string|function}

null (default)
If a string is defined, every result item will receive the string as href attribute replacing any {{itemKey}} by the item's va apply an extra operation of "slugify" on the value `{{url|slugify}}`. - Demo
* If this options is used, an "href" key will be added to every objects to be re-used inside the callbacks.

```
// Replaces group and display item values
href: "/location/{{group}}/{{display}}.html"
```

If a function is defined, the function will be given the item as the first param. It is then for you to build a returned strin

```
// Be careful as item properties might contain Url-unsafe characters
href: function (item) {
    return "/location/" + item.group + "/" + item.display + ".html"
}
```

display Advanced {array}

["display"] (default)
The key that will be searched for typeahead matching results inside source objects. It is possible to search through m adding them inside the configuration array.

```
display: ["series", "description"],
source: {
```

```
    data: [
        {
            series: "Breaking Bad",
            description: "A chemistry teacher ...",
            seasons: 5,
            rating: 9.6
        },{
            series: "Game of Thrones",
            description: "Several noble families fight ...",
            seasons: 4,
            rating: 9.5
        },{
            series: "Dexter",
            description: "A Miami police forensics ...",
            seasons: 8,
            rating: 9.0
        }
    ]
}

// Typeahead can also search deep inside an object, just separate the keys with "."
display: ['string', 'deeper.key.level'],
source: {
    data: [{
        string: 'string',
        deeper: {
            key: {
                level: 42
            }
        }
    }]
}
```

template
Advanced

{string|function}

null (default)

The template is a HTML string containing keys that will replaced by match results object keys. You **MUST** use `{{varia` string to be replaced. - Demo 1 - Demo 2

You can also reach multi-level deep object properties using regular "." format, `{{variable.secordlevel.thirdlevel}}`

If you need to print the item values inside HTML data attributes, it is possible to use {{variable|raw}}. That optional m get the unmodified value.

If a function is used, the first parameter will be the "query" and the second will be the current "item"

```
// Only one group of data, only one template
display: "series",
template: '<span data-series="{{series|raw}}">' +
    '{{series}}, {{seasons}} seasons - ' +
    '<var data-rating="{{rating|raw}}">{{rating}}/10</var></span>',
source: {
    data: [
        {
            series: "Breaking Bad",
            seasons: 5,
            rating: 9.6
        }
        ...
    ]
}

// If the item has a high rating, display a special ribbon saying "Top Rated"
// VERY IMPORTANT to return the template string
display: "series",
template: function (query, item) {
    var template = '<span data-series="{{series|raw}}">' +
        '{{series}}, {{seasons}} seasons - ' +
        '<var data-rating="{{rating|raw}}">{{rating}}/10</var></span>'

    if (item.rating >= 9) {
        template += '<span class="ribbon">Top Rated</span>';
    }
    return template;
},
source: {
    data: [
        {
            series: "Breaking Bad",
            seasons: 5,
            rating: 9.6
        }
        ...
    ]
}

// Multiple groups of data, multiple templates
// Since source.series has no "template" key, the global template will be used
// source.movies is using it's own template
// source.movies uses "title" and "franchise" as the searchable keys
display: "series",
template: "<span>{{series}}, {{seasons}} seasons - rating: <var>{{rating}}</var></span>"
source: {
    series: {
        data: [
            {
                series: "Breaking Bad",
                seasons: 5,
                rating: 9.6
            }
            ...
        ]
    },
    movies: {
        display: ["title", "franchise"],
        template: "<span>" +
            "{{title}}, {{ordinal}} of {{total}} movies from" +
            "{{franchise}} franchise with a budget of {{details.budget}}" +
        "</span>",
        data: [
            {
                title: "Terminator 2",
                ordinal: "2nd",
                total: 5,
                franchise: "Terminator",
                details: {
                    budget: '$6,400,000'
                    gross: '$38,400,000'
                }
            }
            ...
        ]
    }
}
```

emptyTemplate

{string|function}

In case there are no results to be displayed, a row will be displayed containing this template. It is possible to display th string.

```
emptyTemplate: "No results found for {{query}}"

// A function can also be used receiving the "query" param
// IMPORTANT: return a string
emptyTemplate: function (query) {
    if (query.length > 0) {
        return 'No results found for "' + query + '"';
    }
}

// It is possible to return a jQuery <LI> object
emptyTemplate: function (query) {
    return $('<li>', {
        "text": "just use \"" + query + "\"",
        "class": "my-custom-class"
    });
}
```

templateValue
Advanced

{string|function}

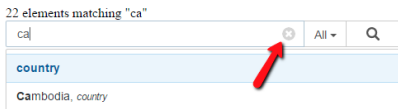
null (default)

When an item is selected / clicked, by default the "Matched key" will go inside the input. By defining a `templateValue`, the input can be customized.

```
// Only one group of data, only one template
display: ["series", "description", "actors.name"],
template: '<span data-series="{{series|raw}}">' +
    '{{series}}, {{seasons}} seasons - ' +
    '<var data-rating="{{rating|raw}}">{{rating}}/10</var></span>',
templateValue: "{{series}}",
source: {
    data: [
        {
            series: "Breaking Bad",
            description: "Breaking Bad is an American crime drama television series created and produced
            actors: [{
```

```
                name: '...',
                age: '...'
            }]
            seasons: 5,
            rating: 9.6
        }
        ...
    ]
}
```

cancelButton	{boolean}	**true** (default) This option provides a small "x" on the right side of the input to clear the text, similar to some browser's default beha `input[type="search"]`.



filter	{boolean|function}	**true** (default) If set to **false**, Typeahead will skip any kind of filtering. This option is useful if you already filter the results in Backend. If set to **function**, every element will be filtered using this custom rule. **WARNING** `accent`, `correlativeTemplate`, `offset` and `matcher` will be skipped

```
// item: The matched item (object)
// displayKey: The matched item display key (string)
filter: function (item, displayKey) {
    if (item.name === "Typeahead") {
        return undefined;
    } else if (/hello/.test(displayKey)) {
        return false;
    } else if (item.name === "hi") {
        // return modified item object
        item.name = "new name key";
        return item;
    }
    return true;
}

// return undefined to skip to next item
// return false to attempt the matching function on the next displayKey
// return true to add the item to the result list
// return item object to modify the item and add it to the result list
```

matcher	{function}	If set to **function**, every element will be filtered using this custom rule AFTER the regular Typeahead filters have been

```
// item: The matched item (object)
// displayKey: The matched item display key (string)
matcher: function (item, displayKey) {
    if (item.name === "Typeahead") {
        return undefined;
    } else if (/hello/.test(displayKey)) {
        return false;
    } else if (item.name === "hi") {
        // return modified item object
        item.name = "new name key";
        return item;
    }
    return true;
}

// return undefined to skip to next item
// return false to attempt the matching function on the next displayKey
// return true to add the item to the result list
// return item object to modify the item and add it to the result list
```

correlativeTemplate	{boolean|array}	**false** (default) By default, search text matching is reserved to `display` keys. A searched text can't match multiple keys. If the option is enabled with `true` or `array` of display keys, every item will reveive an additional key called `compiled` searched first (using soften matching mechanism) for any matching results, then the `display` keys will be searched (matching mechanism). If the option is set to true, the `template` option will be compiled into the "compiled" item key. The search mechanism any word, anywhere separated by spaces. It is also possible to set an Array of display keys instead of the complete template Try it on Hockey_v1
source Required	{object|array}	**null** (default) The source option corresponds to the data set(s) that Typeahead will look through to find matches for the user query Inside the source, you can have multiple lists of data (groups) It is possible to send an async request inside the data function using `$.Deferred` `source.group` configuration:

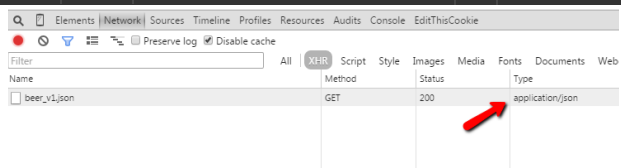
- **minLength**: Overrides the default configuration for the specified group.
- **maxLength**: Overrides the default configuration for the specified group.
- **dynamic**: Overrides the default configuration for the specified group.
- **filter**: Overrides the default configuration for the specified group.
- **matcher**: Overrides the default configuration for the specified group.
- **cache**: Overrides the default configuration for the specified group.
- **compression**: Overrides the default configuration for the specified group.
- **ttl**: Overrides the default configuration for the specified group.
- **data**: Array or function that returns an Array. The items in your array can either be strings or objects
- **href**: Overrides the default configuration for the specified group
- **template**: Overrides the default configuration for the specified group
- **display**: Overrides the default configuration for the specified group
- **ajax**: Local or Jsonp Ajax request to get the data from, follow jquery $.ajax standards.
- **ajax.callback**: Allows `done`, `fail`, `always` & `then` custom callbacks, see example below
- **ajax.data**: Specify the json path to the result array inside the Ajax request

```
// Ajax response ex:
{
    status: 200,
    response: {
        data: {
            user: [{...}, {...}, {...}]
        }
    }
}

// Then the configuration should be:
source: {
    users: {
        ajax: {
            url: '/users/',
            data: {
                q: '{{query}}'
            },
            path: 'response.data.user'
        }
    }
}
```

Source configuration examples:

```
// Simple list of strings
source: ["item1", "item2", "item3"]

// Array of objects, the complete object will be returned inside the callbacks
// Typeahead will look inside the "display" key to match the query string.
// You can change options.display and set it to "author"
source: [{
    id: 1,
    author: "john",
    display: "item1"
},{
    id: 2,
    author: "eric",
    display: "item2"
},{
    id: 3,
    author: "carter",
    display: "item3"
}]


// The source contains 2 groups (game & category)
// source.game "data" and "url" will be combined
// source.game.url is an Array, the data fetched from index[0] and reached by index[1]
// source.category.url is a String, meaning there is only the data set in the response
// Matcher function for EVERY sources
matcher: function (item, displayKey) {},
source: {
    game: {
        data: [{
            id: 1,
            display: "Just an extra game"
        }],
        ajax: {
            url: "/game/list.json",
            path: "data.games"
        }
    },
    category: {
        // Matcher function for "category" only
        matcher: function (item, displayKey) {},
        ajax: {
            method: "GET",
            url: "/category/list.json"
            ...
        }
    }
}

// The Ajax request will receive an object to override the default Ajax request
// Custom done callback is added to execute extra operation on the data
// It is also possible to use fail, then and always callbacks
// Add a custom Authorization Bearer header request
// Every sources will not be filtered by Typeahead, all backend data will appear inside the results
filter: false,
source: {
    teams: {
        // Only "teams" group will not be filtered by Typeahead
        filter: false,
        ajax: {
            type: "POST",
            url: "/jquerytypeahead/hockey_v1.json",
            path: "optional.json.object.index",
            data: {
                myKey: "myValue",
            },
            beforeSend: function (jqXHR, options) {
                jqXHR.setRequestHeader('Authorization', 'Bearer xxxxxxxxxxx');
            },
            // Optional callbacks...
            callback: {
                done: function (data, textStatus, jqXHR) {
                    console.log(data);
                    // Perform operations on received data...
                    // IMPORTANT: data has to be returned if this callback is used
                    return data;
                },
                fail: function (jqXHR, textStatus, errorThrown) {},
                always :function (data, textStatus, jqXHR) {},
                then: function (jqXHR, textStatus) {}
            }
        }
    }
}

// Advanced Ajax request (dynamic) (see user_v1 demo)
// source.user.url[0].data.q "{{query}}" will send the query inside the request
// source.user.data will be merged with the results of the Ajax request
// #user-type updated value gets sent upon request so it bypasses the request cache.
// href creates a link by replacing "{{objetKey}}"
dynamic: true,
source: {
    user: {
        href: "https://www.github.com/{{username}}",
        data: [{
            "id": 415849,
            "username": "an inserted user that is not inside the database",
            "avatar": "https://avatars3.githubusercontent.com/u/415849"
        }],
        ajax: {
            type: "POST",
            // The {{query}} can also be replaced inside the URL
            // url: "/jquerytypeahead/{{query}}",
            url: "/jquerytypeahead/user_v1.json",
            path: "data.user",
            data: {
                q: "{{query}}",
                type: function() { return $('#user-type').val(); }
            }
        }
    }
}

// Set a function that return a request object to have "dynamic" conditions
dynamic: true,
source: {
    tag: {
        ajax: function (query) {
            if (query === "hey") {
                query = "hi"
            }
            return {
                url: "http://www.gamer-hub.com/tag/list.json",
                dataType: "jsonp",
                path: data,
                data: {
                    q: query
                }
            }
        }
    }
}

// Async request using the data function
source: {
    teams: {
        data: function () {

            var deferred = $.Deferred();

            // Gather your data, and resolve the deferred object with an array of objects
            // setTimeout is used only to show that the data is async
            setTimeout(function () {
                if (Array.isArray(data) {
                    deferred.resolve(data);
                } else {
                    deferred.reject("Invalid data.");
                }
            }, 2000);

            deferred.always(function () {
                console.log('yay! :D')
            });

            return deferred;
        }
    }
}
```

* If you are using `url`, make sure the headers of the response is **application/json**.



selector Advanced	{object}	This object contains the CSS classes of all the Typeahead containers. This option will be used to override classes on el your website's UI elements. • **container** `.typeahead__container` (default) Class that is used as the main Typeahead container. This container will also receive `.result`, `.hint`, `.backdrop`, class when the specific feature is toggled to allow for extra CSS customization. • **result** `typeahead__result` (default) Class that will be applied to the result list `div`. • **list** `typeahead__list` (default) Class that will be applied to the result list `div > ul`. • **group** `typeahead__group` (default) Class that will be applied to the group list item if the `group: true` option is set. • **item** `typeahead__item` (default) Class that will be applied to the result list item • **empty** `typeahead__empty` (default) Class that will be applied to the empty list item if `emptyTemplate` is enabled • **display** `typeahead__display` (default) Class that will be applied to the result list `div > ul > li > a`. • **query** `typeahead__query` (default) Block that contains the search input. • **filter** `typeahead__filter` (default) Block that contains the filter. • **filterButton** `typeahead__filter-button` (default) The button inside the block that contains the filter. • **dropdown** `typeahead__dropdown` (default) The dropdown container inside the filters. • **dropdownItem** `typeahead__dropdown-item"` (default) The form submit button container. • **button** `typeahead__button` (default) Block that contains the submit button • **backdrop** `typeahead__backdrop` (default) Class that will be applied to the backdrop container if the `backdrop: true` option is set. • **hint** `typeahead__hint` (default) Class that will be applied to the hint input `hint: true` option is set (should have the same class as your input so t • **cancelButton** `typeahead__cancel-button` (default) Class applied on the cancel button if the option is activated

Callbacks

The callbacks are used to customize and add functionality to your Typeahead instance. You will find plenty of examples in the Demo section.
There are 3 ways to define the callbacks:

- Function [recommended]

Anonymous function calling a local function with parameters.

```
$('#myInput').typeahead({
    source: [ ... ]
    callback: {
        onResult: function (node, query, result,
            console.log(node, query, result, resu
        }
    }
});
```

- String

Function name (can be namespaced) located on the window scope without any parameters.

```
$('#myInput').typeahead({
    source: [ ... ]
    callback: {
        onInit: 'myFunction'
    }
});
```

OR

```
$('#myInput').typeahead({
    source: [ ... ]
    callback: {
        onInit: 'window.myCollection.myFunction.
    }
});
```

- Array

First element is the function name **accessible from the window scope**, second element is an array containing the parameters.

```
$('#myInput').typeahead({
    source: [ ... ]
    callback: {
        onClickBefore: ['window.myClass.myFunction', ['param1', 'param2']]
    }
});

window.myClass = {
    // Your params will be Prepended to the regular Typeahead onClickBefore params
    myFunction: function (param1, param2, node, a, item, event) {
        console.log(param1, param2, node, a, item, event);
    }
};
```

- `node` jQuery object of the initialization input
- `query` The query string inside the input
- `lis` jQuery list of li's
- `a` jQuery object of the item's "a" inside the result list
- `item` Matched item from the result list
- `result` Array of matched items
- `resultCount` Total number of results that matches the query (regardless of the maxItem option)

- `resultHtmlList` HTML <ul> list of the query-matched results (jQuery object)
- `event` jQuery event that was triggered

Same principle as `onPopulateSource` callback but only occurs ONCE, when the data gets inserted in LocalStorage or SessionStorage using the `cache` option.
* The `data` parameter **HAS** to be returned after it's transformed.

Callback	Description
onInit (node)	Will be executed on Typeahead initialization, before anything else.
onReady (node)	Triggers when the Typeahead initial preparation is completed.
onShowLayout (node, query)	Triggers when the Typeahead results layout is displayed.
onHideLayout (node, query)	Triggers when the Typeahead results layout is requested to hide.
onSearch (node, query)	Triggers every time a new search is executed in Typeahead.
onResult (node, query, result, resultCount, resultCountPerGroup)	Whenever the result changes, this callback will be triggered.

onLayoutBuiltBefore (node, query, result, resultHtmlList)	When the result HTML is build, modify it before it get showed. This callback should be used to modify the result DOM before it gets inserted into Typeahead.
	* If you are using this callback, the resultHtmlList param needs to be returned at the end of your function.
	`// Adding the query as the last item`
	`onLayoutBuiltBefore: function (node, query, result, resultHtmlList) {`
	`  return resultHtmlList.append(`
	`  $('<li/>', {`
	`  "text": "Just use \"" + query + "\"",`
	`  "class": "my-custom-class"`
	`  });`
	`}`
	`// Replace all underscore "_" by spaces in the result list`
	`onLayoutBuiltBefore: function (node, query, result, resultHtmlList) {`
	`  $.each(resultHtmlList.find('a'), function (i, a) {`
	`  a.text = a.text.replace(/_/g, ' ');`
	`  });`
	`  return resultHtmlList;`
	`}`

onLayoutBuiltAfter (node, query, result)	Perform an action right after the result HTML gets inserted into Typeahead's DOM.

onNavigateBefore (node, query, event)	When a key is pressed to navigate the results. It is possible to disable the input text change when using up and down arrows when `event.preventInputChange` is set to true
	`callback: {`
	`  onNavigateBefore: function (node, query, event) {`
	`  if (~[38,40].indexOf(event.keyCode)) {`
	`  event.preventInputChange = true;`
	`  }`
	`  }`
	`}`

onNavigateAfter (node, lis, a, item, query, event)	Called at the end of Navigate (once the `.active` class and other operations are completed).
onMouseEnter (node, a, item, event)	Will be executed when a item is hovered inside the result list.
onMouseLeave (node, a, item, event)	Will be executed when a result item is hovered out.
onClick and onClickBefore (node, a, item, event)	Will be executed when a result item is clicked or the right arrow is pressed when an item is selected from the results list. This function will trigger **before** the regular behaviors.
onClickAfter (node, a, item, event)	Will be executed when a result item is clicked or the right arrow is pressed when an item is selected from the results list. This function will trigger **after** the regular behaviors.
onDropdownFilter (node, a, item, event)	Will be executed when a dropdown filter is selected. Requires `dropdownFilter: true`.
onSendRequest (node, query)	Gets called when the Ajax request(s) are sent. Either on initial requests or on every dynamic requests.
onReceiveRequest (node, query)	Gets called when the Ajax request(s) are all received

onPopulateSource (node, data, group, path)	Gets called after the Ajax requests are all received and the data is populated inside Typeahead. This is the place where extra parsing or filtering should occure before the data gets available inside any Typeahead query
	For example, the Backend sends the "display" key separated by underscores "_" instead of spaces " ".
	* The `data` parameter **HAS** to be returned after it's transformed.
	`callback: {`
	`  onPopulateSource: function (node, data, group, path) {`
	`  // Only apply the change for a specific group`
	`  if (group === "mySourceGroup") {`
	`  for (var i = 0; i < data.length; i++) {`
	`  data[i].compiled = data[i].compiled.replace(/_/g, ' ');`
	`  data[i].display = data[i].compiled.replace(/_/g, ' ');`
	`  }`
	`  }`
	`  return data;`
	`  }`
	`}`

onCacheSave (node, data, group, path)	
onSubmit (node, form, item, event)	Override the native onSubmit function by your own. If after performing a set of action(s) you want to submit the form, simply do `form.submit()`. * The `item` parameter is not always defined. An item object will be sent if the submit happens after an item from the list has been selected.
onCancel (node, event)	Any time there is text inside the input and it gets cleared (Backspace, Esc, Cancel button, etc). It is possible to track back the event that cleared the input using `event.originalEvent`

Extras

This section enumerates some functionalities & properties that are accessible

- Typeahead instance properties

* The Typeahead instance uses the same identifier as the selector it was created with.

```
$('#mySearchInput').typeahead({
    ...
});

// Enumerate Typeahead properties
window.Typeahead['#mySearchInput']

rawQuery = '';              // Unmodified input query
query = '';                 // Input query
source = {};                // The generated source kept in memory
isGenerated = null;         // Generated results -> null: not generated, false: generating, true generated
generatedGroupCount = 0;    // Number of groups generated, if limit reached the search can be done
groupCount = 0;             // Number of groups, this value gets counted on the initial source unification
groupBy = "group";          // This option will change according to filtering or custom grouping
result = [];                // Results based on Source-query match (only contains the displayed elements)
resultCount = 0;            // Total results based on Source-query match
options = options;          // Typeahead options (Merged default & user defined)
node = node;                // jQuery object of the Typeahead <input>
container = null;           // Typeahead container, usually right after <form>
resultContainer = null;     // Typeahead result container (html)
item = null;                // The selected item
xhr = {};                   // Ajax request(s) stack
hintIndex = null;           // Numeric value of the hint index in the result list
filters = {                 // Filter list for searching, dropdown and dynamic(s)
    dropdown: {},               // Dropdown menu if options.dropdownFilter is set
    dynamic: {}                 // Checkbox / Radio / Select to filter the source data
};
requests = {};              // Store the group:request instead of generating them every time

backdrop = {};              // The backdrop object
hint = {};                  // The hint object
```

- Triggering manual Typeahead events

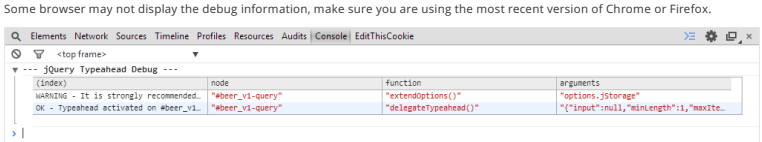
Each of the Typeahead events can be controlled by script and not only by user interaction. They all require a set of condition to be executed. The exact conditions can be tracked down inside the `delegateEvents()` function.

```
// Events
'focus.typeahead',
'input.typeahead',
'propertychange.typeahead',
'keydown.typeahead',
'dynamic.typeahead',
'generateOnLoad.typeahead'

// Trigger a manual search from the string inside the input
// See #Game_v3 demo to understand why it is being used at that place
$('#mySearchInput').trigger('input.typeahead');
```

Debug

Option	Type	Description
debug	{boolean} [optional]	**false** (default) When `debug` configuration is set to **true**, information about the typeahead will be printed in the browser console (F12). If you have difficulties to get the typeahead working on one of your form(s) try to activate the debug mode. * The debug option and outputs are removed from the minified file to reduce its size, so once your configuration is set properly, you should be using **jquery.typeahead.min.js** `$("#myInput").typeahead({ source: [ ... ] debug: true });`

Some browser may not display the debug information, make sure you are using the most recent version of Chrome or Firefox.



Flowchart

The following flowcharts doesn't contain every possibilities and options but only reflects the general code flow.