GAE Atuin CMS Documentation

Release 0.1

Stefano Cilloni, Paolo Casciello, Luca Zarotti, see contributors

Jun 05, 2018

Contents

Version: 0.1

CHAPTER 1

Features

- **Fast.**
- **Easy.** No complex configuration. HTML attributes supported.
- **Modals supported.** No problems in modals.
- **Customizable.** You can customize every single step in the suggesting workflow.
- **Batteries included.** It works out of the box for Bootstrap v3 and v4.
- **i18n.** Use `data-*` attributes to specify the strings to use in case of errors/noresults.
- **Styles.** No custom styles. Uses standard Bootstrap's dropdown.

CHAPTER 2

Getting Started

Bootstrap Autocomplete works as a plugin. Add it to your page

```
<script src="bootstrap-autocomplete.min.js"></script>
```

Using CDN (thanks to RawGit | GitCDN)

Listing 1: STABLE version 0.1

```
<script src="https://cdn.rawgit.com/xcash/bootstrap-autocomplete/v2.0.0/dist/latest/
↪bootstrap-autocomplete.min.js"></script>
```

Listing 2: Latest version (this is the development branch)

```
<script src="https://gitcdn.link/repo/xcash/bootstrap-autocomplete/master/dist/latest/
↪bootstrap-autocomplete.min.js"></script>
```

That's it! Go on to enhance your text fields! :)

CHAPTER 3

Basic usage

3.1 Text Autocomplete

Autocomplete is not enabled by default. You must activate it on the fields you want to enhance. Of course you can also use a wide selector to enable it on specific classes or tags.

Suppose you have a field as follows

```
<input class="form-control basicAutoComplete" type="text" autocomplete="off">
```

Here the class `basicAutoComplete` is used to identify all the fields on which to activate a basic autocomplete. Then in Javascript we activate it:

```
$('.basicAutoComplete').autoComplete({
    resolverSettings: {
        url: 'testdata/test-list.json'
    }
});
```

In this example we specified the `url` to use. Autocomplete will automatically make an Ajax GET request to that URL using an argument named `q` with the text typed by the user. Rate limits are enforced and minimun field length is 2.

Even simpler you can pass the URL directly in the markup

```
<input class="form-control basicAutoComplete" type="text"
        data-url="myurl"
        autocomplete="off">
```

and enhance it just with

```
$('.basicAutoComplete').autoComplete();
```

3.2 Response Format

We know how to start an autocomplete lookup but what about the results?

The *default* configuration expects a simple list in JSON format. Like

```
[
    "Google Cloud Platform",
    "Amazon AWS",
    "Docker",
    "Digital Ocean"
]
```

3.3 Select Autocomplete

One of the main features of Bootstrap Autocomplete is to enhance `<select>` fields as easy as `<input>` text fields. Selects are useful to **restrict choices** to a set of possibilities.

Enhancing a select is no different than text fields.

```
<select class="form-control basicAutoSelect" name="simple_select"
    placeholder="type to search..."
    data-url="testdata/test-select-simple.json" autocomplete="off"></select>
```

```
$('.basicAutoSelect').autoComplete();
```

Nice! :)

3.4 Response Format for Select

In this case we need two values in the response: an `id` and a `text`.

```
[
    { "value": 1, "text": "Google Cloud Platform" },
    { "value": 2, "text": "Amazon AWS" },
    { "value": 3, "text": "Docker" },
    { "value": 4, "text": "Digital Ocean" }
]
```

3.5 Events

Bootstrap Autocomplete triggers usual events.

`change` - Value changed

And custom.

`autocomplete.select` - (evt, item) The element `item` is the item selected by the user and currently selected in the field.

`autocomplete.freevalue` - (evt, value) The text field contains *value* as the custom value (i.e. not selected from the choices dropdown).

CHAPTER 4

Reference

4.1 Activating Autocomplete

$ *(...).autoComplete([options])*
Enhance the form fields identified by the selector

Arguments

- **`options`** – Configuration options of type ConfigOptions.

4.2 Configuration options

`formatResult`

`callback` (*item*)

Arguments

- **`item`** (*`object`*) – The item selected or rendered in the dropdown.

Returns An object `{ id: myItemId, text: myfancyText, html?: myfancierHtml }`.

`minLength`
Default: `3`. Minimum character length to start lookup.

`autoSelect`
Default: `true`. Automatically selects selected item on *blur event* (i.e. using TAB to switch to next field).

`resolver`
Default: `ajax`. Resolver type. `custom` to implement your resolver using *events*.

`noResultsText`
Default: `No results`. Text to show when no results found.

resolverSettings
Object to specify parameters used by default resolver.

url
Url used by default resolver to perform lookup query.

events
Object to specify custom event callbacks.

search

func(*qry*, *callback*)
Function called to perform a lookup.

Arguments
- **qry** (*string*) – Query string.
- **callback** – Callback function to process results. Called passing the **list** of results `callback(results)`.

searchPost

func(*resultsFromServer*)
Function called to manipulate server response. Bootstrap Autocomplete needs a list of items. Use this function to convert any server response in a list of items without reimplementing the default AJAX server lookup.

Arguments
- **resultsFromServer** – Result received from server. Using the default resolver this is an object.

Returns List of items.

Following events are available to fine tune every lookup aspect. Rarely used in common scenarios

typed

func(*newValue*)
Field value changed. Use this function to change the searched value (like prefixing it with some string, filter some characters, ...). Or to stop lookup for certain values.

Arguments
- **newValue** (*string*) – New value.

Returns (Un)modified value or `false` to stop the execution.

searchPre

func(*newValue*)
Before starting the search. Like in the `typed` event, this function can change the search value. The difference is this event is called *after* minLength checks.

Arguments
- **newValue** (*string*) – New value.

Returns (Un)modified value or `false` to stop the execution.

As a reference the lookup workflow calls events in the following order:

```
typed -> searchPre -> search -> searchPost
```

CHAPTER 5

Advanced usage

5.1 Set custom value

To set an initial or change the value of the field.

```
$('.myAutoSelect').autoComplete('set', { value: myValue, text: myText });
```

5.2 Customize results using default AJAX resolver

Using the `searchPost` event you can manipulate the result set making it compatible with autocomplete default. This is useful to bypass the customization of the entire search AJAX call.

```
$('.myAutoSelect').autoComplete({
    events: {
        searchPost: function (resultFromServer) {
            return resultFromServer.results;
        }
    }
});
```

CHAPTER 6

Demo and Examples

You can view Demo and Examples here.

CHAPTER 7

Translating messages

To customize “no results” message use the following markup.

```
<select class="form-control emptyAutoSelect" name="empty_select"
    data-url="testdata/test-empty.json"
    data-noresults-text="Nothing to see here."
    autocomplete="off"></select>
```

CHAPTER 8

Issues, Support and New Features requests

Feel free to post a new issue here

CHAPTER 9

Development Environment

To setup an environment to develop Bootstrap-Autocomplete you need only Docker and Docker Compose.

The source is in the TypeScript language in the `src` directory while the documentation is generated using Sphinx and resides in the `docs` directory.

To start the environment:

```
$ docker-compose up
```

Two servers starts up:

- Demo page
- Documentation

Index