completer

simple js autocompletion

here is a demo page - you can submit the form and check the request data via inspection i am working on the documentation ;)

if you have any ideas about improvements, found some bugs, please let me know and open a ticket.

about

i needed a simple completer / suggester for one of the forms in a project. in the past i used select2, a quite awesome project. unfortunately this project is based on jquery and i do my projects in vanilla js nowerdays. all other projects i found were buggy, had lacks for configurations.. so i decided to do it on my own..

here is my version of a js autocompleter :)

you instantiate a new configuration object, get guided via fluent setters and pass that configuration to a new instance of the autocompleter class.

You can decide if you want to provide all available data via an array or provide the data behind an api. This script can do a post or a get request. You can specify additional headers that may include an api key or something like that. You can specify the data key, default is 'term'.

As search operations can be expensive, you can choose to cache the results. If you set the cache time to 0, no cache will be created. Otherwise the cache for one term will last the seconds you provide.

You can choose if you want one or more items to be selected via the 'setMaxItemsSelected' method.

If you have more than one element that you want to autocomplete, you can create a base configuration instance and clone it. So you do not have to create and copy a bunch of code.

Of course you can specify translations so your users are provided terms in their language they can understand.

You can specify the name of the key for the value that will be stored in the origin input, that data will be sent to the server in the request.

If you want to display user-friendly data, you can specify two other keys:

- displaySearchKeyOfData -> will the displayed in favour of the value while searching
- displaySelectKeyOfData -> will be displayed in favour of the displaySearchKeyOfData or value after selecting a result

configuration

get a fresh configuration object

```
const conf = new AutocompleteConfiguration()
```

debugging

Sometimes you want to know more information about what happens in the engine. default value: disabled To enable use

```
conf.enableDebug()
```

To disable use

```
conf.disableDebug()
```

Specify element

You can pass the reference to the configuration.

```
conf.setElement(document.getElementById('elementOne'))
```

Specify element ID

You can pass the id of the element to the configuration.

```
conf.setElementId('elementOne')
```

Initial data

If the user visits the page, some data may be selected already. You can set the initial selected data with:

```
conf.setInitialData([
    {value: 'bar'}
])
```

Selection limits

You can specify the maximum amount of selected items. $1 =$ just one $> 1 =$ n elements $< 1 =$ unlimited

```
conf.setMaxItemsSelected(2)
```

Fixed data set

You can set the available data via

```
conf.setData([
    {value: 'foorem'},
    {value: 'barsum'},
    {value: 'red'},
    {value: 'black'},
    {value: 'blue'},
])
```

search url (absolute)

You can specify the absolute url for searching. If you choose to use get requests, the search will be appended to the get-parameter-list. We need to distinguish if the search url is relative or absolute because of possible additional parameters in the url.

```
conf.setAbsoluteSearchUrl('https://foo.bar.com/search/colors?some=data&more=data')
```

search url (relative)

You can specify the relative url for searching. If you choose to use get requests, the search will be appended to the get-parameter-list. We need to distinguish if the search url is relative or absolute because of possible additional parameters in the url.

```
conf.setRelativeSearchUrl('/search/colors?some=data&more=data')
```

search post key

If the script should do an ajax post request, you can specify the key of the data. Default is 'term';

```
conf.setSearchPostKey('search')
```

search get key

If the script should do an ajax get request, you can specify the key of the data. Default is 'term';

```
conf.setSearchGetKey('search')
```

additional headers

Nowadays it is almost mandatory to send some headers to an api to fulfill the request requirements. This may include api keys, origins, or something like that. clear all headers:

```
conf.clearAdditionalHeaders()
```

add a single header:

```
conf
    .addAdditionalHeader('key', 'value')
    .addAdditionalHeader('api-key', 'some-uuid-value')
```

cache results

enable caching for one second:

```
conf.setCacheResultsSeconds(1)
```

enable caching for 20 seconds:

```
conf.setCacheResultsSeconds(20)
```

disable caching:

```
conf.setCacheResultsSeconds(0)
```

delay searching

If you want to delay the requests made to the server / api, you can specify the milliseconds the script waits until the request is made.
The default value is 100.

```
conf.setSearchDelay(0)
```

callbacks

You can provide callbacks that will be called.

On starting a request:

```
conf.setRequestStartCallback(function (query) {
    console.log('searching for ' + query);
})
```

On request is finished:

```
conf.setRequestEndCallback(function (query, results) {
    console.log('searching ended for ' + query);
    console.log(results);
})
```

On selecting a result:

```
conf.setSelectCallback(function (item) {
    console.log('picked result');
    console.log(item);
})
```

translations

You can provide callbacks that will be called.

Placeholder in search input:

```
conf.setTranslationPlaceholder('type to search') //default
conf.setTranslationPlaceholder('Tippen für Suche')
conf.setTranslationPlaceholder('dotknij, aby wyszukać')
conf.setTranslationPlaceholder('appuyez pour rechercher')
```

No results found for search term:

```
conf.setTranslationNoResults('no results') //default
conf.setTranslationNoResults('keine Treffer')
conf.setTranslationNoResults('żadnych trafień')
conf.setTranslationNoResults('aucun succès')
```

Example configuration with provided data

```
/**
* @type {boolean}
*/
this.debugEnabled = false;

        /**
         * the element itself that should be used for autocompletion
         * @type {null|Element}
         */
        this.element = null;

        /**
         * the id of the element that should be used for autocompletion
         * @type {null|string}
         */
        this.elementId = null;

        /**
         * initially selected data
         * @type {string[]}
         */
        this.initialData = [];

        this.maxItemsSelected = 1; // how many items should be maximally selected?

        this.data = null; // if provided, no ajax search is needed

        this.absoluteSearchUrl = null; // if you want ajax search, provide the absolute url
        this.relativeSearchUrl = null; // if you want ajax search, provide the relative url
        this.searchPostKey = 'term'; // the key in the post-request that contains the searc
        this.searchGetKey = 'term'; // the key in the get-request that contains the search p
        this.additionalHeaders = {}; // a list of additional headers, e.g., api key, cookie,

        this.cacheResultsSeconds = 0; // how long results should be cached, 0 for no cache
        this.searchDelay = 100; // wait milliseconds until a search will be made, prevents c

        /**
         * @type {string}
         */
        this.valueKeyOfData = 'value'; // the key of a dataset where the value is stored
        this.displaySearchKeyOfData = null; // will be displayed in the result list instead
        this.displaySelectKeyOfData = null; // will be displayed as the selected value inste

        this.resultContainerClasses = ['resultContainer']; // CSS class of result container
        this.resultContainerId = 'resultContainer'; // the id of the result container

        this.requestStartCallback = null; // a callback when the ajax search is started
        this.requestEndCallback = null; // a callback when the ajax search finished
        this.selectCallback = null; // a callback when a result item was selected

        this.translationNoResults = 'no results'; // the translation for not finding any res
        this.translationPickResult = 'pick result'; // the translation for pick a result
        this.translationClearResults = 'clear results'; // the translation for clear all res
        this.translationMoreResults = 'more results'; // the translation for search for more
        this.translationErrorMessage = 'an error occurred'; // the translation if an error o
```

```
<script src="/js/autocompletion.js"></script>
```

```
console.log("test");
document.addEventListener('DOMContentLoaded', function () {
    const conf = new AutocompleteConfiguration()
        .enableDebug()
        .setMaxItemsSelected(1)
        .setDisplaySelectKeyOfData('display')
        .setDisplaySearchKeyOfData('search')
        .setCacheResultsSeconds(10)
        .setSearchGetKey('query')
    ;
    new Autocomplete(
        conf
            .clone()
            .setRelativeSearchUrl('/search/foo')
            .setElement(document.getElementById('elementOne'))
    );
    new Autocomplete(
        conf
            .clone()
            .setRelativeSearchUrl('/search/foo')
            .setElement(document.getElementById('elementTwo'))
    );
});
```