()

Select2 3.5.2 (/select2/index.html) Select2 2.1 (/select2/select-2.1.html) Select2 1.0 (/select2/select2-1.0.html)

Select2 Latest (/select2/select2-latest.html)

Fork me on GitHub

Select2 3.5.2

Select2 is a jQuery based replacement for select boxes. It supports searching, remote data sets, and infinite scrolling of results.

Learn more on GitHub» (https://github.com/ivaynberg/select2) Download» (https://github.com/ivaynberg/select2/tags)

Star Fork 2,322 Follow @ivaynberg

Changelog Examples Documentation About

Changelog 3.5.2 (8)

show/hide details

Browser Compatibility

- IE 8+
- Chrome 8+
- Firefox 10+
- Safari 3+
- Opera 10.6+

Examples

The Basics

Select2 can take a regular select box like this:

Alaska

and turns it into:

Alaska

with support for quick option filtering via a search box.

Example Code

```
<head>
    <link href="select2.css" rel="stylesheet"/>
    <script src="select2.js"></script>
    <script>
        $(document).ready(function() { $("#e1").select2(); });
    </script>
</head>
<body>
    <select id="e1">
        <option value="AL">Alabama</option>
        ...
        <option value="WY">Wyoming</option>
    </select>
</body>
```

Multi-Value Select Boxes

Select2 also supports multi-value select boxes. The `select` below is declared with the `multiple` attribute. Select2 automatically picks up on this:

Example Code

```
1.  $("#e9").select2();
```

Placeholders

A placeholder value can be defined and will be displayed until a selection is made:

Select a State

Select a State

The placeholder can be declared via a `data-placeholder` attribute attached to the `select`, or via the `placeholder` configuration element as seen in the example code.

When placeholder is used for a non-multi-value select box, it requires that you include an empty `<option></option>` tag as your first option.

Optionally, a clear button (visible once a selection is made) is available to reset the select box back to the placeholder value.

Example Code

```
1.  $("#e2").select2({
2.      placeholder: "Select a State",
3.      allowClear: true
4.  });
5.  $("#e2_2").select2({
6.      placeholder: "Select a State"
7.  });
```

Minimum Input

Select2 supports a minimum input setting which is useful for large remote datasets where short search terms are not very useful:

Alaska

Example Code

```
1.  $("#e3").select2({
2.      minimumInputLength: 2
3.  });
```

Templating

Various display options of the Select2 component can be changed:

Alaska

Example Code

```
1.  function format(state) {
2.      if (!state.id) return state.text; // optgroup
3.      return "<img class='flag' src='images/flags/" + state.id.toLowerC
    ase() + ".png'/>" + state.text;
4.  }
5.  $("#e4").select2({
6.      formatResult: format,
7.      formatSelection: format,
8.      escapeMarkup: function(m) { return m; }
9.  });
```

You can set `data-` attributes to `<option>` (or <optgroup>) and use them inside templating functions:

```
1.  <select>
2.      <option value="0" data-foo="bar">option one</option>
3.      ...
4.  </select>
5.
```

```
function format(state) {
    var originalOption = state.element;

    return "<img class='flag' src='images/flags/" + state.id.toLowerCase() + ".png' alt='" + $(originalOption).data('foo') + "' />" + state.text;
}
```

Loading Data

Select2 uses a function to load result data. Here is a trivial example that creates choices that consist of user's input repeated a number of times:

In order to take advantage of custom data loading Select2 should be attached to an `input type='hidden'` tag, otherwise data is parsed from `select`'s `option` tags.

Example Code

```
$("#e5").select2({
    minimumInputLength: 1,
    query: function (query) {
        var data = {results: []}, i, j, s;
        for (i = 1; i < 5; i++) {
            s = "";
            for (j = 0; j < i; j++) {s = s + query.term;}
            data.results.push({id: query.term + i, text: s});
        }
        query.callback(data);
    }
});
```

Maximum Selection Size

Select2 allows the developer to limit the number of items that can be selected in a multi-select control. In the example below only 3 or less items can be selected.

Example Code

```
$("#e19").select2({ maximumSelectionSize: 3 });
```

Loading Array Data

Select2 provides some shortcuts that make it easy to access local data stored in an array instead of having to write a `query` function mentioned in the example above.

Example below inlines the data by specifying an array in the `data` element. Items in such an array must have `id` and `text` keys.

If your data does not have a `text` key, an alternative key can be specified as a string:

or as a function:

`data` can also itself be a function that returns a results object:

Example Code

```
$("#e10").select2({
    data:[{id:0,text:'enhancement'},{id:1,text:'bug'},{id:2,text:'duplicate'},{id:3,text:'invalid'},{id:4,text:'wontfix'}]
});

var data=[{id:0,tag:'enhancement'},{id:1,tag:'bug'},{id:2,tag:'duplicate'},{id:3,tag:'invalid'},{id:4,tag:'wontfix'}];
function format(item) { return item.tag; }

$("#e10_2").select2({
    data:{ results: data, text: 'tag' },
    formatSelection: format,
    formatResult: format
});

$("#e10_3").select2({
    data:{ results: data, text: function(item) { return item.tag; } },
    formatSelection: format,
    formatResult: format
});
$("#e10_4").select2({
    data:function() { return { text:'tag', results: data }; },
    formatSelection: format,
    formatResult: format
});
```

Loading Remote Data

Select2 comes with AJAX/JSONP support built in. In this example we will search for repositories using Github's API:

ivaynberg/select2

If this example stops working, you have most likely reached the usage limit for the GitHub Search API of 5 requests per minute. Please wait a few minutes and try again.

Example Code

```
$("#e6").select2({
    placeholder: "Search for a repository",
    minimumInputLength: 1,
    ajax: { // instead of writing the function to execute the request we use Select2's convenient helper
        url: "https://api.github.com/search/repositories",
        dataType: 'json',
        quietMillis: 250,
        data: function (term, page) {
            return {
                q: term, // search term
            };
        },
        results: function (data, page) { // parse the results into the format expected by Select2.
            // since we are using custom formatting functions we do not need to alter the remote JSON data
            return { results: data.items };
        },
        cache: true
    },
    initSelection: function(element, callback) {
        // the input tag has a value attribute preloaded that points to a preselected repository's id
        // this function resolves that id attribute to an object that select2 can render
        // using its formatResult renderer - that way the repository name is shown preselected
        var id = $(element).val();
        if (id !== "") {
            $.ajax("https://api.github.com/repositories/" + id, {
                dataType: "json"
            }).done(function(data) { callback(data); });
        }
    },
    formatResult: repoFormatResult, // omitted for brevity, see the source of this page
    formatSelection: repoFormatSelection,  // omitted for brevity, see the source of this page
    dropdownCssClass: "bigdrop", // apply css that makes the dropdown taller
    escapeMarkup: function (m) { return m; } // we do not want to escape markup since we are displaying html in results
});
```

Select2 uses jQuery's `$.ajax` function to execute the remote call by default. An alternative `transport` function can be specified in the ajax settings, or an entirely custom implementation can be built by providing a custom `query` function instead of using the `ajax` helper.

Infinite Scroll with Remote Data

Select2 supports lazy-appending of results when the result list is scrolled to the end. In order to enable the remote service must support some sort of a paging mechanism and the query function given to Select2 must take advantage of it. The following example demonstrates how this can be set up. Search for some keyword and then scroll the result list to the end to see more results load:

Search for a repository

If this example stops working, you have most likely reached the usage limit for the GitHub Search API of 5 requests per minute. Please wait a few minutes and try again.

Example Code

```
$("#e7").select2({
    placeholder: "Search for a repository",
    minimumInputLength: 3,
    ajax: {
        url: "https://api.github.com/search/repositories",
        dataType: 'json',
        quietMillis: 250,
        data: function (term, page) { // page is the one-based page number tracked by Select2
            return {
                q: term, //search term
                page: page // page number
            };
        },
        results: function (data, page) {
            var more = (page * 30) < data.total_count; // whether or not there are more results available

            // notice we return the value of more so Select2 knows if more results can be loaded
            return { results: data.items, more: more };
        }
    },
    formatResult: repoFormatResult, // omitted for brevity, see the source of this page
    formatSelection: repoFormatSelection, // omitted for brevity, see the source of this page
    dropdownCssClass: "bigdrop", // apply css that makes the dropdown taller
    escapeMarkup: function (m) { return m; } // we do not want to escape markup since we are displaying html in results
});
```

Programmatic Access

Select2 supports methods that allow programmatic control of the component:

Alert selected value | Set to California | Clear | Alert selected using data | Set to California using data | Open | Close

Alaska

Alert selected value | Set to California and Massachusetts | Alert selected value using data | Set to California and Massachusetts using data | Clear | Open | Close

Select a state

Example Code

```
$("#e8").select2();
$("#e8_get").click(function () { alert("Selected value is: "+$("#e8").select2("val"));});
$("#e8_set").click(function () { $("#e8").select2("val", "CA"); });
$("#e8_cl").click(function() { $("#e8").select2("val", ""); });
$("#e8_get2").click(function () { var data = $("#e8").select2("data"); delete data.element; alert("Selected data is: "+JSON.stringify(data));});
$("#e8_set2").click(function () { $("#e8").select2("data", {id: "CA", text: "California"}); });
$("#e8_open").click(function () { $("#e8").select2("open"); });
$("#e8_close").click(function () { $("#e8").select2("close"); });
$("#e8_2").select2({placeholder: "Select a state"});
$("#e8_2_get").click(function () { alert("Selected value is: "+$("#e8_2").select2("val"));});
$("#e8_2_set").click(function () { $("#e8_2").select2("val", ["CA","MA"]); });
$("#e8_2_get2").click(function () { alert("Selected value is: "+JSON.stringify($("#e8_2").select2("data")));});
$("#e8_2_set2").click(function () { $("#e8_2").select2("data", [{id: "CA", text: "California"},{id:"MA", text: "Massachusetts"}]); });
$("#e8_2_cl").click(function() { $("#e8_2").select2("val", ""); });
$("#e8_2_open").click(function () { $("#e8_2").select2("open"); });
$("#e8_2_close").click(function () { $("#e8_2").select2("close"); });
```

Events

`change` event is triggered on the original element whenever its value is changed by the user

`open` event is triggered on the original element whenever the dropdown needs to be opened:

Event Log

Example Code

Select report type

```
$("#e11").select2({
  placeholder: "Select report type",
  allowClear: true,
  data: [{id: 0, text: 'story'},{id: 1, text: 'bug'},{id: 2, text: 'task'}]
});
$("#e11_2").select2({
    createSearchChoice:function(term, data) { if ($(data).filter(function() { return this.text.localeCompare(term)===0; }).length===0) {return {id:term, text:term};} },
    multiple: true,
    data: [{id: 0, text: 'story'},{id: 1, text: 'bug'},{id: 2, text: 'task'}]
});
function log(e) {
    var e=$("<li>"+e+"</li>");
    $("#events_11").append(e);
    e.animate({opacity:1}, 10000, 'linear', function() { e.animate({opacity:0}, 2000, 'linear', function() {e.remove(); }); });
}
$("#e11")
    .on("change", function(e) { log("change "+JSON.stringify({val:e.val, added:e.added, removed:e.removed})); })
    .on("select2-opening", function() { log("opening"); })
    .on("select2-open", function() { log("open"); })
    .on("select2-close", function() { log("close"); })
    .on("select2-highlight", function(e) { log ("highlighted val="+ e.val+" choice="+ JSON.stringify(e.choice));})
    .on("select2-selecting", function(e) { log ("selecting val="+ e.val+" choice="+ JSON.stringify(e.choice));})
    .on("select2-removing", function(e) { log ("removing val="+ e.val+" choice="+ JSON.stringify(e.choice));})
    .on("select2-removed", function(e) { log ("removed val="+ e.val+" choice="+ JSON.stringify(e.choice));})
    .on("select2-loaded", function(e) { log ("loaded (data property omitted for brevity)");})
    .on("select2-focus", function(e) { log ("focus");})
    .on("select2-blur", function(e) { log ("blur");});
$("#e11_2")
    .on("change", function(e) { log("change "+JSON.stringify({val:e.val, added:e.added, removed:e.removed})); })
    .on("select2-opening", function() { log("opening"); })
    .on("select2-open", function() { log("open"); })
    .on("select2-close", function() { log("close"); })
    .on("select2-highlight", function(e) { log ("highlighted val="+ e.val+" choice="+ JSON.stringify(e.choice));})
    .on("select2-selecting", function(e) { log ("selecting val="+ e.val+" choice="+ JSON.stringify(e.choice));})
    .on("select2-removing", function(e) { log ("removing val="+ e.val+" choice="+ JSON.stringify(e.choice));})
    .on("select2-removed", function(e) { log ("removed val="+ e.val+" choice="+ JSON.stringify(e.choice));})
    .on("select2-loaded", function(e) { log ("loaded (data property omitted for brevity)");})
    .on("select2-focus", function(e) { log ("focus");})
    .on("select2-blur", function(e) { log ("blur");});
```

Tagging Support

Select2 can be used to quickly set up fields used for tagging.

brown red green

Note that when tagging is enabled the user can

Example Code

```
$("#e12").select2({tags:["red", "green", "blue"]});
```

select from pre-existing tags or create a new tag by picking the first choice which is what the user has typed into the search box so far.

Maximum Input Length

Select2 can be set a limit on the number of characters that can be entered per tag.

brown red green

You would not be able to enter any input of more than 10 characters long.

Example Code

```
1.                    $("#e23").select2({
2.                        tags:["red", "green", "blue"],
3.                        maximumInputLength: 10
4.                    });
```

Auto Tokenization

Select2 supports ability to add choices automatically as the user is typing into the search field. This is especially convenient in the tagging usecase where the user can quickly enter a number of tags by separating them with a comma or a space. Try typing in the search field below and entering a space or a comma.

brown

Note that the separators are defined in the tokenSeparators option.

Note that this example uses the built in tokenizer function, but a custom one can be provided in the options.

Example Code

```
1.                    $("#e20").select2({
2.                        tags:["red", "green", "blue"],
3.                        tokenSeparators: [",", " "]});
```

Reacting to external value changes

Select2 can react to external value changes and keep its selection in-sync. This feature allows Select2 to work seamlessly with front-end frameworks that use data binding between ui components and model values.

Select California

Select Alaska and Colorado

This feature is only available when initSelection() function is provided in the options. This function is needed to map the choice ids set on the element to objects used by Select2. This function is set by default when Select2 is attached to a `select` or when the `tags` helper function is used.

Example Code

```
1.  $("#e13").select2();
2.  $("#e13_ca").click(function() { $("#e13").val("CA").trigger("change")
    ; });
3.  $("#e13_ak_co").click(function() { $("#e13").val(["AK","CO"]).trigger
    ("change"); });
```


Select2 Lifecycle

Example Code

Alabama

Init Destroy

```
$("#e14").val(["AL","AZ"]).select2();
$("#e14_init").click(function() { $("#e14").select2(); });
$("#e14_destroy").click(function() { $("#e14").select2("destroy"); });
```

Select2 Drag and Drop Sorting

Select2 supports drag and drop sorting of selected choices. Select2 does not, itself, provide the necessary code to perform dragging and dropping, instead it provides hooks that other libraries can use to provide the behavior. In this example we are using JQuery UI's `sortable()` plugin.

The sorting is only available when Select2 is attached to a hidden `input` field.

red green blue orange white black

Example Code

```
$("#e15").select2({tags:["red", "green", "blue", "orange", "white", "black", "purple", "cyan", "teal"]});
$("#e15").on("change", function() { $("#e15_val").html($("#e15").val());});

$("#e15").select2("container").find("ul.select2-choices").sortable({
    containment: 'parent',
    start: function() { $("#e15").select2("onSortStart"); },
    update: function() { $("#e15").select2("onSortEnd"); }
});
```

Select2 Disabled Mode

Alaska

Enable Disable

Writable Readonly

Example Code

```
$("#e16").select2();
$("#e16_2").select2();
$("#e16_enable").click(function() { $("#e16,#e16_2").select2("enable", true); });
$("#e16_disable").click(function() { $("#e16,#e16_2").select2("enable", false); });
$("#e16_readonly").click(function() { $("#e16,#e16_2").select2("readonly", true); });
$("#e16_writable").click(function() { $("#e16,#e16_2").select2("readonly", false); });
```

Custom Matcher

Unlike other dropdowns on this page, this one matches options only if the term appears in the beginning of the string as opposed to anywhere:

Alaska

The dropdown below matches on custom attributes of the `option` tag. For example, the `blue` option can be matched by entering either `blue` or `cyan`:

red

Example Code

```
$("#e17").select2({
      matcher: function(term, text) { return text.toUpperCase().indexOf(term.toUpperCase())==0; }
});
// <select id="e17_2" style="width:300px">
//     <option alt="pink">red</option>
//     <option alt="cyan">blue</option>
// </select>
$("#e17_2").select2({
   matcher: function(term, text, opt) {
       return text.toUpperCase().indexOf(term.toUpperCase())>=0
           || opt.attr("alt").toUpperCase().indexOf(term.toUpperCase())>=0;
   }
});
```


Sorting Displayed

Example Code

Results

Unlike other dropdowns on this page, this one filters results by query string normally, but returns the matched results sorted from shortest to longest by string length. Try typing 'e' and seeing how the results are sorted. This function is useful for sorting results by relevance to a user's query.

red

```
$('#e22').select2({
    sortResults: function(results, container, query) {
        if (query.term) {
            // use the built in javascript sort function
            return results.sort(function(a, b) {
                if (a.text.length > b.text.length) {
                    return 1;
                } else if (a.text.length < b.text.length) {
                    return -1;
                } else {
                    return 0;
                }
            });
        }
        return results;
    }
});
```

Responsive Design - Percent Width

Select2's width can be set to a percentage of its parent to support responsive design. The two Select2 boxes below are styled to 50% and 75% width respectively.

Alaska

Select a state

Select2 will do its best to resolve the percent width specified via a css class, but it is not always possible. The best way to ensure that Select2 is using a percent based width is to inline the style declaration into the tag.

Example Code

```
$("#e18,#e18_2").select2();
```

Disabled options

In case that you need to disable certain options so that they can't be selected by the select2 interface, you can now pass in `disabled: true` with your data. Please note: This also works for incoming values from ajax. When attached to a `<select>` element, Select2 will respect its `<option>` elements' `disabled` property.

Example Code

```
$(document).ready(function () {
    $('#e24').select2({
        query: function (query){
            var data = {
                results: [
                    { id: 1, text: "I'm selectable" },
                    { id: 2, text: "I'm a disabled option", disabled: true },
                    { id: 3, text: "I'm selectable too!" }
                ]
            };

            query.callback(data);
        }
    });
```

Lock selections

In the event that you need to lock certain selections so that they can't be removed by the select2 interface, you can now pass in `locked: true` with your data. Please note: This also works for incoming values from ajax.

Disabled User | Jane Doe | John Doe | Robert Paulson | Spongebob Squarepants | Planet Bob | Inigo Montoya

Example Code

```
var preload_data = [
  { id: 'user0', text: 'Disabled User', locked: true}
  , { id: 'user1', text: 'Jane Doe'}
  , { id: 'user2', text: 'John Doe', locked: true }
  , { id: 'user3', text: 'Robert Paulson', locked: true }
  , { id: 'user5', text: 'Spongebob Squarepants'}
  , { id: 'user6', text: 'Planet Bob' }
  , { id: 'user7', text: 'Inigo Montoya' }
];

$(document).ready(function () {
  $('#e21').select2({
      multiple: true
      ,query: function (query){
          var data = {results: []};

          $.each(preload_data, function(){
              if(query.term.length == 0 || this.text.toUpperCase().indexOf(query.term.toUpperCase()) >= 0 ){
                  data.results.push({id: this.id, text: this.text });
              }
          });

          query.callback(data);
      }
  });
  $('#e21').select2('data', preload_data )
```

Diacritics Support

Select2's default matcher will ignore diacritics, making it easier for users to filter results in international selects. Type "aero" into the select below:

Documentation

Constructor

Parameter	Type	Description

width	string	Controls the `width` style attribute of the Select2 container div. The following values are supported: **off** No width attribute will be set. Keep in mind that the container div copies classes from the source element so setting the width attribute may not always be necessary. **element** Uses javascript to calculate the width of the source element. **copy** Copies the value of the width style attribute set on the source element. **resolve** First attempts to copy than falls back on element. **other values** if the width attribute contains a function it will be evaluated, otherwise the value is used verbatim.
minimumInputLength	int	Number of characters necessary to start a search.
maximumInputLength	int	Maximum number of characters that can be entered for an input.
minimumResultsForSearch	int	The minimum number of results that must be initially (after opening the dropdown for the first time) populated in order to keep the search field. This is useful for cases where local data is used with just a few results, in which case the search box is not very useful and wastes screen space. The option can be set to a `negative value` to permanently hide the search field. Only applies to single-value select boxes
maximumSelectionSize	int/function	The maximum number of items that can be selected in a multi-select control. If this number is less than 1 selection is not limited. Once the number of selected items reaches the maximum specified the contents of the dropdown will be populated by the `formatSelectionTooBig` function.
placeholder	string	Initial value that is selected if no other selection is made. The placeholder can also be specified as a `data-placeholder` attribute on the `select` or `input` element that Select2 is attached to. Note that because browsers assume the first `option` element is selected in non-multi-value select boxes an empty first `option` element must be provided (`<option></option>`) for the placeholder to work.
placeholderOption	function/string	When attached to a `select` resolves the `option` that should be used as the placeholder. Can either be a function which given the `select` element should return the `option` element or a string `first` to indicate that the first option should be used. This option is useful when Select2's default of using the first option only if it has no value and no text is not suitable.
separator	string	Separator character or string used to delimit ids in `value` attribute of the multi-valued selects. The default delimiter is the `,` character.

allowClear	boolean	Whether or not a clear button is displayed when the select box has a selection. The button, when clicked, resets the value of the select box back to the placeholder, thus this option is only available when the placeholder is specified. This option only works when the placeholder is specified. When attached to a `select` an `option` with an empty value must be provided. This is the option that will be selected when the button is pressed since a select box requires at least one selection `option`. Also, note that this option only works with non-multi-value based selects because multi-value selects always provide such a button for every selected option.
multiple	boolean	Whether or not Select2 allows selection of multiple values. When Select2 is attached to a `select` element this value will be ignored and `select`'s `multiple` attribute will be used instead.
closeOnSelect	boolean	If set to false the dropdown is not closed after a selection is made, allowing for rapid selection of multiple items. By default this option is set to `true`. Only applies when configured in multi-select mode.
openOnEnter	boolean	If set to true the dropdown is opened when the user presses the enter key and Select2 is closed. By default this option is enabled.
id	function	Function used to get the id from the choice object or a string representing the key under which the id is stored. `id(object)` **Parameter** / **Type** / **Description** object / object / A choice object. <returns> / string / the id of the object. The default implementation expects the object to have a `id` property that is returned.

<table>
<tr>
<td>matcher</td>
<td>function</td>
<td>
Used to determine whether or not the search term matches an option when a built in query function is used. The built in query function is used when Select2 is attached to a <code>select</code>, or the <code>local</code> or <code>tags</code> helpers are used.
<pre>matcher(term, text, option)</pre>
<table>
<tr><th>Parameter</th><th>Type</th><th>Description</th></tr>
<tr><td>term</td><td>string</td><td>search term.</td></tr>
<tr><td>text</td><td>string</td><td>text of the option being matched.</td></tr>
<tr><td>option</td><td>jquery object</td><td>the <code>option</code> element we are trying to match. Only given when attached to <code>select</code>. Can be used to match against custom attributes on the <code>option</code> tag in addition to matching on the <code>option</code>'s text.</td></tr>
<tr><td><returns></td><td>boolean</td><td><code>true</code> if search term matches the text, or <code>false</code> otherwise.</td></tr>
</table>
The default implementation is case insensitive and matches anywhere in the term:
<code>function(term, text) { return text.toUpperCase().indexOf(term.toUpperCase())>=0; }</code>
</td>
</tr>
<tr>
<td>sortResults</td>
<td>function</td>
<td>
Used to sort the results list for searching right before display. Useful for sorting matches by relevance to the user's search term.
<pre>sortResults(results, container, query)</pre>
.
<table>
<tr><td>object</td><td>object</td><td>One of the result objects returned from the <code>query</code> function</td></tr>
<tr><td>container</td><td>jQuery object</td><td>jQuery wrapper of the node that should contain the representation of the result.</td></tr>
<tr><td>query</td><td>object</td><td>The query object used to request this set of results.</td></tr>
<tr><td><returns></td><td>object</td><td>A results object.</td></tr>
</table>
Defaults to no sorting: <code>function(results, container, query) { return results; }</code>
</td>
</tr>
</table>

formatSelection | function | Function used to render the current selection.

```
formatSelection(object, container)
```

Parameter	Type	Description
object	object	The selected result object returned from the `query` function.
container	jQuery object	jQuery wrapper of the node to which the selection should be appended.
escapeMarkup	function	Function that can be used to escape html markup. This is the function defined in the `escapeMarkup` option, or the default.
<returns>	string (optional)	Html string, a DOM element, or a jQuery object that renders the selection.

The default implementation expects the object to have a `text` property that is returned.

The implementation may choose to append elements directly to the provided `container` object, or return a single value and have it automatically appended.

When attached to a `select` the original `<option>` (or <optgroup>) element is accessible inside the specified function through the property `item.element`:

```
format(item) {
    var originalOption = item.element;
    return item.text
}
```

formatResult | function

Function used to render a result that the user can select.

```
formatResult(object, container, query)
```

Parameter	Type	Description
object	object	One of the result objects returned from the `query` function.
container	jQuery object	jQuery wrapper of the node that should contain the representation of the result.
query	object	The query object used to request this set of results.
escapeMarkup	function	Function used to escape markup in results. If you do not expect to render custom markup you should pass your text through this function to escape any markup that may have been accidentally returned. This function is configurable in options of select2.
<returns>	string (optional)	Html string, a DOM element, or a jQuery object that represents the result.

The default implementation expects the object to have a `text` property that is returned.

The implementation may choose to append elements directly to the provided `container` object, or return a single value and have it automatically appended.

When attached to a `select` the original `<option>` (or <optgroup>) element is accessible inside the specified function through the property `item.element`:

```
format(item) {
    var originalOption = item.element;
    return item.text
}
```

formatResultCssClass | function

Function used to add css classes to result elements.

```
formatResultCssClass(object)
```

Parameter	Type	Description
object	object	One of the result objects returned from the `query` function.
<returns>	string (optional)	String containing css class names separated by a space.

By default when attached to a `select` css classes from `option`s will be automatically copied.

<table>
<tr>
<td>formatNoMatches</td>
<td>string/function</td>
<td>String containing "No matches" message, or
Function used to render the message
<pre>formatNoMatches(term)</pre>
<table>
<tr><th>Parameter</th><th>Type</th><th>Description</th></tr>
<tr><td>term</td><td>string</td><td>Search string entered by user.</td></tr>
<tr><td><returns></td><td>string</td><td>Message html.</td></tr>
</table>
</td>
</tr>
<tr>
<td>formatSearching</td>
<td>string/function</td>
<td>String containing "Searching..." message, or
Function used to render the message that is displayed while search is in progress.
<pre>formatSearching()</pre>
<table>
<tr><th>Parameter</th><th>Type</th><th>Description</th></tr>
<tr><td><returns></td><td>string</td><td>Message html or <code>null</code> / <code>undefined</code> to disable the message.</td></tr>
</table>
</td>
</tr>
<tr>
<td>formatAjaxError</td>
<td>string/function</td>
<td>String containing "Loading Failed" message, or
Function used to render the message
<pre>formatAjaxError(jqXHR, textStatus, errorThrown)</pre>
<table>
<tr><th>Parameter</th><th>Type</th><th>Description</th></tr>
<tr><td>jqXHR</td><td>object</td><td>The XMLHttpRequest object.</td></tr>
<tr><td>textStatus</td><td>string</td><td>Ajax request status text.</td></tr>
<tr><td>errorThrown</td><td>string</td><td>Server response status text.</td></tr>
<tr><td><returns></td><td>string</td><td>Message html.</td></tr>
</table>
For more info on the parameters, refer to the JQuery API Documentation (http://api.jquery.com/jquery.ajax/).
</td>
</tr>
<tr>
<td>formatInputTooShort</td>
<td>string/function</td>
<td>String containing "Search input too short" message, or
Function used to render the message.
<pre>formatInputTooShort(term, minLength)</pre>
<table>
<tr><th>Parameter</th><th>Type</th><th>Description</th></tr>
<tr><td>term</td><td>string</td><td>Search string entered by user.</td></tr>
<tr><td>minLength</td><td>int</td><td>Minimum required term length.</td></tr>
<tr><td><returns></td><td>string</td><td>Message html.</td></tr>
</table>
</td>
</tr>
</table>

formatInputTooLong | string/function | String containing "Search input too long" message, or Function used to render the message.

```
formatInputTooLong(term, maxLength)
```

Parameter	Type	Description
term	string	Search string entered by user.
maxLength	int	Maximum required term length.
<returns>	string	Message html.

formatSelectionTooBig | string/function | String containing "You cannot select any more choices" message, or Function used to render the message.

```
formatSelectionTooBig(maxSize)
```

Parameter	Type	Description
maxSize	string	The maximum specified size of the selection.
<returns>	string	Message html.

formatLoadMore | string/function | String containing "Loading more results…" message, or Function used to render the message.

```
formatLoadMore(pageNumber)
```

Parameter	Type	Description
pageNumber	string	The current page.
<returns>	string	Message html.

createSearchChoice | function | Creates a new selectable choice from user's search term. Allows creation of choices not available via the query function. Useful when the user can create choices on the fly, eg for the 'tagging' usecase.

```
createSearchChoice(term)
```

Parameter	Type	Description
term	string	Search string entered by user.
<returns>	object (optional)	Object representing the new choice. Must at least contain an `id` attribute.

If the function returns `undefined` or `null` no choice will be created. If a new choice is created it is displayed first in the selection list so that user may select it by simply pressing `enter`.

When used in combination with `input[type=hidden]` tag care must be taken to sanitize the `id` attribute of the choice object, especially stripping `,` as it is used as a value separator.

createSearchChoicePosition	string|function	Define the position where to insert element created by `createSearchChoice`. The following values are supported: **top** Insert in the top of the list **bottom** Insert at the end of the list **<function>** A custom function. For example if you want to insert the new item in the second position: `$("#tags").select2({ ... createSearchChoice: function(term) { ... }, createSearchChoicePosition: function(list, item) { list.splice(1, 0, item); } });`

initSelection	function	Called when Select2 is created to allow the user to initialize the selection based on the value of the element select2 is attached to.
		Essentially this is an `id->object` mapping function.
		
		`initSelection(element, callback)`
		
		Parameter / **Type** / **Description**
		element / jQuery array / element Select2 is attached to.
		callback / function / callback function that should be called with the data which is either an object in case of a single select or an array of objects in case of multi-select.
		
		This function will only be called when there is initial input to be processed.
		
		Here is an example implementation used for tags. Tags are the simplest form of data where the id is also the text:
		
		`$("#tags").select2({ initSelection : function (element, callback) { var data = []; $(element.val().split(",")).each(function () { data.push({id: this, text: this}); }); callback(data); } }); // Or for single select elements: $("#select").select2({ initSelection : function (element, callback) { var data = {id: element.val(), text: element.val()}; callback(data); } });`

<table>
<tr>
<td>tokenizer</td>
<td>function</td>
<td>
A tokenizer function can process the input typed into the search field after every keystroke and extract and select choices. This is useful, for example, in tagging scenarios where the user can create tags quickly by separating them with a comma or a space instead of pressing enter.

Tokenizer only applies to multi-selects.

<code>tokenizer(input, selection, selectCallback, opts)</code>

<table>
<tr><th>Parameter</th><th>Type</th><th>Description</th></tr>
<tr><td>input</td><td>string</td><td>The text entered into the search field so far.</td></tr>
<tr><td>selection</td><td>array</td><td>Array of objects representing the current selection. Useful if tokenizer needs to filter out duplicates.</td></tr>
<tr><td>selectCallback</td><td>function</td><td>Callback that can be used to add objects to the selection.</td></tr>
<tr><td>opts</td><td>object</td><td>Options with which Select2 was initialized. Useful if tokenizer needs to access some properties in the options.</td></tr>
<tr><td><returns></td><td>string (optional)</td><td>Returns the string to which the input of the search field should be set to. Usually this is the remainder, of any, of the string after the tokens have been stripped. If <code>undefined</code> or <code>null</code> is returned the input of the search field is unchanged.</td></tr>
</table>

The default tokenizer will only be used if the <code>tokenSeparators</code> and <code>createSearchChoice</code> options are specified. The default tokenizer will split the string using any separator in <code>tokenSeparators</code> and will create and select choice objects using <code>createSearchChoice</code> option. It will also ignore duplicates, silently swallowing those tokens.
</td>
</tr>
<tr>
<td>tokenSeparators</td>
<td>array</td>
<td>An array of strings that define token separators for the default tokenizer (doc-tokenizer) function. By default, this option is set to an empty array which means tokenization using the default tokenizer is disabled. Usually it is sensible to set this option to a value similar to <code>[',', ' ']</code>.</td>
</tr>
</table>

query	function	Function used to query results for the search term. `query(options)` **Parameter** / **Type** / **Description** options.element — jquery object — The element Select2 is attached to. options.term — string — Search string entered by user. options.page — int — 1-based page number tracked by Select2 for use with infinite scrolling of results. options.context — object — An object that persists across the lifecycle of queries for the same search term (the query to retrieve the initial results, and subsequent queries to retrieve more result pages for the same search term). When this function is first called for a new search term this object will be null. The user may choose to set any object in the `results.context` field - this object will then be used as the context parameter for all calls to the `query` method that will load more search results for the initial search term. The object will be reset back to null when a new search term is queried. This feature is useful when a page number is not easily mapped against the server side paging mechanism. For example, some server side paging mechanism may return a "continuation token" that needs to be passed back to them in order to retrieve the next page of search results. options.callback — function — Callback function that should be called with the `result` object. The result object: **Parameter** / **Type** / **Description** result.results — [object] — Array of result objects. The default renderers expect objects with `id` and `text` keys. The `id` property is required, even if custom renderers are used. The object may also contain a `children` key if hierarchical data is displayed. The object may also contain a `disabled` boolean property indicating whether this result can be selected. result.more — boolean — `true` if more results are available for the current search term. results.context — object — A user-defined object that should be made available as the `context` parameter to the `query` function on subsequent queries to load more result pages for the same search term. See the description of `options.context` parameter. In order for this function to work Select2 should be attached to a `input type="hidden"`

tag instead of a `select`.

Example Data

```
{
     more: false,
     results: [
        { id: "CA", text: "California" },
        { id: "AL", text: "Alabama" }
     ]
}
```

Example Hierarchical Data

```
{
    more: false,
    results: [
        { text: "Western", children: [
            { id: "CA", text: "California" },
            { id: "AZ", text: "Arizona" }
        ] },
        { text: "Eastern", children: [
            { id: "FL", text: "Florida" }
        ] }
    ]
}
```

ajax	object	Options for the built in ajax query function. This object acts as a shortcut for having to manually write a function that performs ajax requests. The built-in function supports more advanced features such as throttling and dropping out-of-order responses.

Parameter	Type	Description
transport	function	Function that will be used to perform the ajax request. Must be parameter-compatible with `$.ajax`. Defaults to `$.ajax` if not specified. Allows the use of various ajax wrapper libraries such as: AjaxManager (http://www.protofunc.com/scripts/jquery/ajaxManager/).
url	string/function	String containing the ajax url or a function that returns such a string.
dataType	string	Data type for the request. `xml`, `json`, `jsonp`, other formats supported by jquery.
quietMillis	int	Number of milliseconds to wait for the user to stop typing before issuing the ajax request.
cache	boolean	If set to `false`, it will force requested pages not to be cached by the browser. Default is `false`.
jsonpCallback	string/function	The callback function name for a JSONP request. This value will be used instead of the random name automatically generated by jQuery. It is preferable to let jQuery generate a unique name as it'll make it easier to manage the requests and provide callbacks and error handling. You may want to specify the callback when you want to enable better browser caching of GET requests.
data	function	Function to generate query parameters for the ajax request.

```
data(term, page)
```

Parameter	Type	Description
term	string	Search term.
page	int	1-based page number tracked by Select2 for use with infinite scrolling of results.
context	object	See `options.context` parameter to the `query` function above.
<returns>	object	Object containing url parameters.

<table>
<tr><td></td><td></td><td>
<table>
<tr><td>results</td><td>function</td><td>Function used to build the query results object from the ajax response
<code>results(data, page, query)</code>
<table><tr><th>Parameter</th><th>Type</th><th>Description</th></tr><tr><td>data</td><td>object</td><td>Retrieved data.</td></tr><tr><td>page</td><td>int</td><td>Page number that was passed into the <code>data</code> function above.</td></tr><tr><td>query</td><td>object</td><td>The query object used to request this set of results.</td></tr><tr><td><returns></td><td>object</td><td>Results object. See "options.callback" in the "query" function for format.</td></tr></table></td></tr>
<tr><td>params</td><td>object/function</td><td>An object or a function that returns an object that contains extra parameters that will be passed to the transport. For example it can be used to set the content type:
<code>{contentType: "application/json;charset=utf-8"}</code></td></tr>
</table>
In order for this function to work Select2 should be attached to a <code>input type='hidden'</code> tag instead of a <code>select</code>.
For documentation of the data format see the query function.
</td></tr>
<tr><td>data</td><td>array/object</td><td>Options for the built in query function that works with arrays.
If this element contains an array, each element in the array must contain <code>id</code> and <code>text</code> keys.
Alternatively, this element can be specified as an object in which <code>results</code> key must contain the data as an array and a <code>text</code> key can either be the name of the key in data items that contains text or a function that retrieves the text given a data element from the array.</td></tr>
<tr><td>tags</td><td>array/function</td><td>Puts Select2 into 'tagging'mode where the user can add new choices and pre-existing tags are provided via this options attribute which is either an <code>array</code> or a <code>function</code> that returns an array of <code>objects</code> or <code>strings</code>. If <code>strings</code> are used instead of <code>objects</code> they will be converted into an object that has an <code>id</code> and <code>text</code> attribute equal to the value of the <code>string</code>.</td></tr>
<tr><td>containerCss</td><td>function/object</td><td>Inline css that will be added to select2's container. Either an object containing css property/value key pairs or a function that returns such an object.</td></tr>
<tr><td>containerCssClass</td><td>function/string</td><td>Css class that will be added to select2's container tag.</td></tr>
<tr><td>dropdownCss</td><td>function/object</td><td>Inline css that will be added to select2's dropdown container. Either an object containing css property/value key pairs or a function that returns such an object.</td></tr>
<tr><td>dropdownCssClass</td><td>function/string</td><td>Css class that will be added to select2's dropdown container.</td></tr>
<tr><td>dropdownAutoWidth</td><td>boolean</td><td>When set to <code>true</code> attempts to automatically size the width of the dropdown based on content inside.</td></tr>
</table>

<table>
<tr><td>adaptContainerCssClass</td><td>function</td><td>Function that filters/renames css classes as they are copied from the source tag to the select2 container tag.
<code>adaptContainerCssClass(clazz)</code>
<table><tr><th>Parameter</th><th>Type</th><th>Description</th></tr><tr><td>clazz</td><td>string</td><td>Css class being copied.</td></tr><tr><td><returns></td><td>string</td><td>Css class to be applied or <code>null/undefined/''</code> to not apply it.</td></tr></table>The default implementation applies all classes without modification.</td></tr>
<tr><td>adaptDropdownCssClass</td><td>function</td><td>Function that filters/renames css classes as they are copied from the source tag to the select2 dropdown tag.
<code>adaptDropdownCssClass(clazz)</code>
<table><tr><th>Parameter</th><th>Type</th><th>Description</th></tr><tr><td>clazz</td><td>string</td><td>Css class being copied.</td></tr><tr><td><returns></td><td>string</td><td>Css class to be applied or <code>null/undefined/''</code> to not apply it.</td></tr></table>The default implementation always returns <code>null</code> thereby filtering out all classes.</td></tr>
<tr><td>escapeMarkup</td><td>function</td><td><code>String escapeMarkup(String markup)</code>
Function used to post-process markup returned from formatter functions. By default this function escapes html entities to prevent javascript injection.</td></tr>
<tr><td>selectOnBlur</td><td>boolean</td><td>Set to <code>true</code> if you want Select2 to select the currently highlighted option when it is blurred.</td></tr>
<tr><td>loadMorePadding</td><td>integer</td><td>Defines how many pixels need to be below the fold before the next page is loaded. The default value is <code>0</code> which means the result list needs to be scrolled all the way to the bottom for the next page of results to be loaded. This option can be used to trigger the load sooner, possibly resulting in a smoother user experience.</td></tr>
</table>

nextSearchTerm	function	Function used to determine what the next search term should be. **Parameter** / **Type** / **Description** data / object / Retrieved data. this.search.val() / string / Search term that yielded the current result set. Here is an example implementation used to display the current search term when the dropdown is opened: `function displayCurrentValue(selectedObject, currentSearchTerm) { return currentSearchTerm; } $("#e1").select2({ nextSearchTerm: displayCurrentValue });` Function can be used when the dropdown is configured in single and multi-select mode. It is triggered after selecting an item. In single mode it is also triggered after initSelection (when provided).

val

Gets or sets the selection. If the `value` parameter is not specified, the `id` attribute of the currently selected element is returned. If the `value` parameter is specified it will become the current selection.

Parameter	Type	Description
value (optional)	object	**Attached to `select`**: Single-Valued: Value of the `value` attribute of the `option` that should be selected. Multi-Valued: Array of the `value` attributes of the `option`s that should be selected. `null` for empty. **Attached to `input[type=hidden]`**: Single-Valued: Id of the object that should be selected. `""` to clear. Can only be used if `initSelection()` was specified. Multi-Valued: An array of objects ids that should be selected. `""` to clear. Can only be used if `initSelection()` was specified.
triggerChange (optional)	boolean	Whether or not a `change` event should be triggered. `false` by default.

`val` method invoked on a single-select with an unset value will return `""`, while a `val` method invoked on an empty multi-select will return `[]`.

Example:

```
alert("Selected value is: "+$("#e8").select2("val")); $("#e8").select2("val", "CA");
```

Notice that in order to use this method you must define the `initSelection` function in the options so Select2 knows how to transform the id of the object you pass in `val()` to the full object it needs to render selection. If you are attaching to a `select` element this function is already provided for you.

data

Gets or sets the selection. Analogous to `val` method, but works with objects instead of ids.

`data` method invoked on a single-select with an unset value will return `null`, while a `data` method invoked on an empty multi-select will return `[]`.

destroy

Reverts changes to DOM done by Select2. Any selection done via Select2 will be preserved.

open

Opens the dropdown.

close

Closes the dropdown.

enable(boolean)

Enables or disables Select2 and its underlying form component based on the boolean parameter.

readonly(boolean)

Toggles readonly mode on Select2 and its underlying form component based on the boolean parameter.

container

Retrieves the main container element that wraps all of DOM added by Select2 Example: `console.log($("#tags").select2("container"));`

onSortStart

Notifies Select2 that a drag and drop sorting operation has started. Select2 will hide all non-selection list items such as the search container, etc. Example: `$("#tags").select2("onSortStart");`

onSortEnd

Notifies Select2 that a drag and drop sorting operation has finished. Select2 will re-display any elements previously hidden and update the selection of the element it is attached to. Example: `$("#tags").select2("onSortEnd");`

search

Executes a new search using the provided value. Example: `$("#tags").select2("search","California");`

Events

change

Fired when selection is changed.

The event object contains the following custom properties:

val
The current selection (taking into account the result of the change) - id or array of ids.
added

The added element, if any - the full element object, not just the id.

removed

The removed element, if any - the full element object, not just the id.

select2-opening

Fired before the dropdown is shown.

The event listener can prevent the opening by calling `preventDefault()` on the supplied event object.

select2-open

Fired after the dropdown is shown.

select2-close

Fired after the dropdown is closed.

select2-highlight

Fired when a choice is highlighted in the dropdown.

The event object contains the following custom properties:

val

The id of the highlighted choice object.

choice

The highlighted choice object.

select2-selecting

Fired when a choice is being selected in the dropdown, but before any modification has been made to the selection. This event is used to allow the user to reject selection by calling `event.preventDefault()`

The event object contains the following custom properties:

val

The id of the highlighted choice object.

choice

The choice object about to be selected.

select2-clearing

Fired when a choice is being cleared in the dropdown, but before any modification has been made to the selection. This event is used to allow the user to reject the clear by calling `event.preventDefault()`

For the clear button to be visible the `allowClear` option needs to be `true`.

select2-removing

Fired when a choice is about to be removed in the dropdown/input, but before any removal of the choice has been made. This event is used to allow the user to reject removal by calling `event.preventDefault()`

The event object contains the following custom properties:

val

The id of the removing choice object.

choice

The choice object about to be removed.

select2-removed

Fired when a choice is removed or cleared.

The event object contains the following custom properties:

val
The id of the highlighted choice object.
choice
The highlighted choice object.

select2-loaded

Fired when query function is done loading the data and the results list has been updated

The event object contains the following custom properties:

items
data that was used to populate the results.

select2-focus

Fired when the control is focussed.

select2-blur

Fired when the control is blurred.

Configuring Defaults

Select2 exposes its default options via the `$.fn.select2.defaults` object. Properties changed in this object (same properties configurable through the constructor) will take effect for every instance created after the change.

About

- Project Site (https://github.com/ivaynberg/select2)
- Bug Tracker (https://github.com/ivaynberg/select2/issues)
- Wiki containing example integrations and usages such as Knockout.js, Socket.io, and PHP (https://github.com/ivaynberg/select2/wiki)
- Mailing List (https://groups.google.com/d/forum/select2)
