* Overview
* Options
* Events
* Methods
* Theming

jQuery UI Autocomplete

## Overview

Autocomplete, when added to an input field, enables users to quickly find and select from a pre-populated list of values as they type, leveraging searching and filtering.

By giving an Autocomplete field focus or entering something into it, the plugin starts searching for entries that match and displays a list of values to choose from. By entering more characters, the user can filter down the list to better matches.

This can be used to enter previous selected values, for example you could use Autocomplete for entering tags, to complete an address, you could enter a city name and get the zip code, or maybe enter email addresses from an address book.

You can pull data in from a local and/or a remote source: Local is good for small data sets (like an address book with 50 entries), remote is necessary for big data sets, like a database with hundreds or millions of entries to select from.

Autocomplete can be customized to work with various data sources, by just specifying the source option. A data source can be:

* an Array with local data
* a String, specifying a URL
* a Callback

The local data can be a simple Array of Strings, or it contains Objects for each item in the array, with either a label or value property or both. The label property is displayed in the suggestion menu. The value will be inserted into the input element after the user selected something from the menu. If just one property is specified, it will be used for both, eg. if you provide only value-properties, the value will also be used as the label.

When a String is used, the Autocomplete plugin expects that string to point to a URL resource that will return JSON data. It can be on the same host or on a different one (must provide JSONP). The request parameter "term" gets added to that URL. The data itself can be in the same format as the local data described above.

The third variation, the callback, provides the most flexibility, and can be used to connect any data source to Autocomplete. The callback gets two arguments:

* A request object, with a single property called "term", which refers to the value currently in the text input. For example, when the user entered "new yo" in a city field, the Autocomplete term will equal "new yo".
* A response callback, which expects a single argument to contain the data to suggest to the user. This data should be filtered based on the provided term, and can be in any of the formats described above for simple local data (String-Array or Object-Array with label/value/both properties). It's important when providing a custom source callback to handle errors during the request. You must always call the response callback even if you encounter an error. This ensures that the widget always has the correct state.

The label is always treated as text, if you want the label to be treated as html you can use [Scott González' html extension](https://github.com/scottgonzalez/jquery-ui-extensions/blob/master/autocomplete/jquery.ui.autocomplete.html.js). The demos all focus on different variations of the source-option - look for the one that matches your use case, and take a look at the code.

### Dependencies

* UI Core
* UI Widget
* UI Position

### Example

* Demo
* View Source

A simple jQuery UI Autocomplete.

$("input#autocomplete").autocomplete({  
 source: ["c++", "java", "php", "coldfusion", "javascript", "asp", "ruby"]  
});

<!DOCTYPE html>  
<html>  
<head>  
 <link href="http://ajax.googleapis.com/ajax/libs/jqueryui/1.8/themes/base/jquery-ui.css" rel="stylesheet" type="text/css"/>  
 <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.5/jquery.min.js"></script>  
 <script src="http://ajax.googleapis.com/ajax/libs/jqueryui/1.8/jquery-ui.min.js"></script>  
   
 <script>  
 $(document).ready(function() {  
 $("input#autocomplete").autocomplete({  
 source: ["c++", "java", "php", "coldfusion", "javascript", "asp", "ruby"]  
});  
 });  
 </script>  
</head>  
<body style="font-size:62.5%;">  
   
<input id="autocomplete" />  
  
</body>  
</html>

## Options

#### **disabled**Type: Boolean Default: false Disables (true) or enables (false) the autocomplete. Can be set when initialising (first creating) the autocomplete. Code examplesInitialize a autocomplete with the disabled option specified. $( ".selector" ).autocomplete({ disabled: true });Get or set the disabled option, after init. //getter var disabled = $( ".selector" ).autocomplete( "option", "disabled" ); //setter $( ".selector" ).autocomplete( "option", "disabled", true );

#### **appendTo**Type: Selector Default: "body" Which element the menu should be appended to. Code examplesInitialize a autocomplete with the appendTo option specified. $( ".selector" ).autocomplete({ appendTo: "#someElem" });Get or set the appendTo option, after init. //getter var appendTo = $( ".selector" ).autocomplete( "option", "appendTo" ); //setter $( ".selector" ).autocomplete( "option", "appendTo", "#someElem" );

#### **autoFocus**Type: Boolean Default: false If set to true the first item will be automatically focused. Code examplesInitialize a autocomplete with the autoFocus option specified. $( ".selector" ).autocomplete({ autoFocus: true });Get or set the autoFocus option, after init. //getter var autoFocus = $( ".selector" ).autocomplete( "option", "autoFocus" ); //setter $( ".selector" ).autocomplete( "option", "autoFocus", true );

#### **delay**Type: Integer Default: 300 The delay in milliseconds the Autocomplete waits after a keystroke to activate itself. A zero-delay makes sense for local data (more responsive), but can produce a lot of load for remote data, while being less responsive. Code examplesInitialize a autocomplete with the delay option specified. $( ".selector" ).autocomplete({ delay: 0 });Get or set the delay option, after init. //getter var delay = $( ".selector" ).autocomplete( "option", "delay" ); //setter $( ".selector" ).autocomplete( "option", "delay", 0 );

#### **minLength**Type: Integer Default: 1 The minimum number of characters a user has to type before the Autocomplete activates. Zero is useful for local data with just a few items. Should be increased when there are a lot of items, where a single character would match a few thousand items. Code examplesInitialize a autocomplete with the minLength option specified. $( ".selector" ).autocomplete({ minLength: 0 });Get or set the minLength option, after init. //getter var minLength = $( ".selector" ).autocomplete( "option", "minLength" ); //setter $( ".selector" ).autocomplete( "option", "minLength", 0 );

#### **position**Type: Object Default: { my: "left top", at: "left bottom", collision: "none" } Identifies the position of the Autocomplete widget in relation to the associated input element. The "of" option defaults to the input element, but you can specify another element to position against. You can refer to the [jQuery UI Position](http://docs.jquery.com/UI/Position) utility for more details about the various options. Code examplesInitialize a autocomplete with the position option specified. $( ".selector" ).autocomplete({ position: { my : "right top", at: "right bottom" } });Get or set the position option, after init. //getter var position = $( ".selector" ).autocomplete( "option", "position" ); //setter $( ".selector" ).autocomplete( "option", "position", { my : "right top", at: "right bottom" } );

#### **source**Type: String, Array, Callback Default: none, must be specified Defines the data to use, must be specified. See Overview section for more details, and look at the various demos. Code examplesInitialize a autocomplete with the source option specified. $( ".selector" ).autocomplete({ source: ["c++", "java", "php", "coldfusion", "javascript", "asp", "ruby"] });Get or set the source option, after init. //getter var source = $( ".selector" ).autocomplete( "option", "source" ); //setter $( ".selector" ).autocomplete( "option", "source", ["c++", "java", "php", "coldfusion", "javascript", "asp", "ruby"] );

## Events

#### **create**Type: autocompletecreate This event is triggered when autocomplete is created. Code examplesSupply a callback function to handle the create event as an init option. $( ".selector" ).autocomplete({ create: function(event, ui) { ... } });Bind to the create event by type: autocompletecreate. $( ".selector" ).bind( "autocompletecreate", function(event, ui) { ... });

#### **search**Type: autocompletesearch Before a request (source-option) is started, after minLength and delay are met. Can be canceled (return false), then no request will be started and no items suggested. Code examplesSupply a callback function to handle the search event as an init option. $( ".selector" ).autocomplete({ search: function(event, ui) { ... } });Bind to the search event by type: autocompletesearch. $( ".selector" ).bind( "autocompletesearch", function(event, ui) { ... });

#### **open**Type: autocompleteopen Triggered when the suggestion menu is opened. Code examplesSupply a callback function to handle the open event as an init option. $( ".selector" ).autocomplete({ open: function(event, ui) { ... } });Bind to the open event by type: autocompleteopen. $( ".selector" ).bind( "autocompleteopen", function(event, ui) { ... });

#### **focus**Type: autocompletefocus Before focus is moved to an item (not selecting), ui.item refers to the focused item. The default action of focus is to replace the text field's value with the value of the focused item, though only if the focus event was triggered by a keyboard interaction. Canceling this event prevents the value from being updated, but does not prevent the menu item from being focused. Code examplesSupply a callback function to handle the focus event as an init option. $( ".selector" ).autocomplete({ focus: function(event, ui) { ... } });Bind to the focus event by type: autocompletefocus. $( ".selector" ).bind( "autocompletefocus", function(event, ui) { ... });

#### **select**Type: autocompleteselect Triggered when an item is selected from the menu; ui.item refers to the selected item. The default action of select is to replace the text field's value with the value of the selected item. Canceling this event prevents the value from being updated, but does not prevent the menu from closing. Code examplesSupply a callback function to handle the select event as an init option. $( ".selector" ).autocomplete({ select: function(event, ui) { ... } });Bind to the select event by type: autocompleteselect. $( ".selector" ).bind( "autocompleteselect", function(event, ui) { ... });

#### **close**Type: autocompleteclose When the list is hidden - doesn't have to occur together with a change. Code examplesSupply a callback function to handle the close event as an init option. $( ".selector" ).autocomplete({ close: function(event, ui) { ... } });Bind to the close event by type: autocompleteclose. $( ".selector" ).bind( "autocompleteclose", function(event, ui) { ... });

#### **change**Type: autocompletechange Triggered when the field is blurred, if the value has changed; ui.item refers to the selected item. Code examplesSupply a callback function to handle the change event as an init option. $( ".selector" ).autocomplete({ change: function(event, ui) { ... } });Bind to the change event by type: autocompletechange. $( ".selector" ).bind( "autocompletechange", function(event, ui) { ... });

## Methods

### **destroy**Signature: .autocomplete( "destroy" ) Remove the autocomplete functionality completely. This will return the element back to its pre-init state.

### **disable**Signature: .autocomplete( "disable" ) Disable the autocomplete.

### **enable**Signature: .autocomplete( "enable" ) Enable the autocomplete.

### **option**Signature: .autocomplete( "option" , optionName , [value] ) Get or set any autocomplete option. If no value is specified, will act as a getter.

### **option**Signature: .autocomplete( "option" , options ) Set multiple autocomplete options at once by providing an options object.

### **widget**Signature: .autocomplete( "widget" ) Returns the .ui-autocomplete element.

### **search**Signature: .autocomplete( "search" , [value] ) Triggers a search event, which, when data is available, then will display the suggestions; can be used by a selectbox-like button to open the suggestions when clicked. If no value argument is specified, the current input's value is used. Can be called with an empty string and minLength: 0 to display all items.

### **close**Signature: .autocomplete( "close" ) Close the Autocomplete menu. Useful in combination with the search method, to close the open menu.

## Theming

The jQuery UI Autocomplete plugin uses the jQuery UI CSS Framework to style its look and feel, including colors and background textures. We recommend using the ThemeRoller tool to create and download custom themes that are easy to build and maintain.

If a deeper level of customization is needed, there are widget-specific classes referenced within the jquery.ui.autocomplete.css stylesheet that can be modified. These classes are highlighed in bold below.

### Sample markup with jQuery UI CSS Framework classes

<input class="**ui-autocomplete-input**"/>

<ul class="**ui-autocomplete** **ui-menu** ui-widget ui-widget-content ui-corner-all">

  <li class="**ui-menu-item**">

    <a class="ui-corner-all">item 1</a>

  </li>

  <li class="**ui-menu-item**">

    <a class="ui-corner-all">item 2</a>

  </li>

  <li class="**ui-menu-item**">

    <a class="ui-corner-all">item 3</a>

  </li>

</ul>

**Note: This is a sample of markup generated by the autocomplete plugin, not markup you should use to create a autocomplete. The only markup needed for that is <input/>.**