* Overview
* Options
* Events
* Methods
* Theming

jQuery UI Sortable

## Overview

The jQuery UI Sortable plugin makes selected elements sortable by dragging with the mouse.

All callbacks receive two arguments: The original browser event and a prepared ui object, view below for a documentation of this object (if you name your second argument 'ui'):

* **ui.helper** - the current helper element (most often a clone of the item)
* **ui.position** - current position of the helper
* **ui.offset** - current absolute position of the helper
* **ui.item** - the current dragged element
* **ui.placeholder** - the placeholder (if you defined one)
* **ui.sender** - the sortable where the item comes from (only exists if you move from one connected list to another)

### Dependencies

* UI Core
* UI Widget
* UI Mouse

### Example

* Demo
* View Source

A simple jQuery UI Sortable.

$("#sortable").sortable();

<!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() {  
 $("#sortable").sortable();  
 });  
 </script>  
</head>  
<body style="font-size:62.5%;">  
   
<ul id="sortable">  
<li>Item 1</li>  
<li>Item 2</li>  
<li>Item 3</li>  
<li>Item 4</li>  
<li>Item 5</li>  
</ul>  
  
</body>  
</html>

## Options

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

#### **appendTo**Type: String Default: 'parent' Defines where the helper that moves with the mouse is being appended to during the drag (for example, to resolve overlap/zIndex issues). Code examplesInitialize a sortable with the appendTo option specified. $( ".selector" ).sortable({ appendTo: 'body' });Get or set the appendTo option, after init. //getter var appendTo = $( ".selector" ).sortable( "option", "appendTo" ); //setter $( ".selector" ).sortable( "option", "appendTo", 'body' );

#### **axis**Type: String Default: false If defined, the items can be dragged only horizontally or vertically. Possible values:'x', 'y'. Code examplesInitialize a sortable with the axis option specified. $( ".selector" ).sortable({ axis: 'x' });Get or set the axis option, after init. //getter var axis = $( ".selector" ).sortable( "option", "axis" ); //setter $( ".selector" ).sortable( "option", "axis", 'x' );

#### **cancel**Type: Selector Default: ':input,button' Prevents sorting if you start on elements matching the selector. Code examplesInitialize a sortable with the cancel option specified. $( ".selector" ).sortable({ cancel: 'button' });Get or set the cancel option, after init. //getter var cancel = $( ".selector" ).sortable( "option", "cancel" ); //setter $( ".selector" ).sortable( "option", "cancel", 'button' );

#### **connectWith**Type: Selector Default: false Takes a jQuery selector with items that also have sortables applied. If used, the sortable is now connected to the other one-way, so you can drag from this sortable to the other. Code examplesInitialize a sortable with the connectWith option specified. $( ".selector" ).sortable({ connectWith: '.otherlist' });Get or set the connectWith option, after init. //getter var connectWith = $( ".selector" ).sortable( "option", "connectWith" ); //setter $( ".selector" ).sortable( "option", "connectWith", '.otherlist' );

#### **containment**Type: Element, String, Selector Default: false Constrains dragging to within the bounds of the specified element - can be a DOM element, 'parent', 'document', 'window', or a jQuery selector. Note: the element specified for containment must have a calculated width and height (though it need not be explicit), so for example, if you have float:left sortable children and specify containment:'parent' be sure to have float:left on the sortable/parent container as well or it will have height: 0, causing undefined behavior. Code examplesInitialize a sortable with the containment option specified. $( ".selector" ).sortable({ containment: 'parent' });Get or set the containment option, after init. //getter var containment = $( ".selector" ).sortable( "option", "containment" ); //setter $( ".selector" ).sortable( "option", "containment", 'parent' );

#### **cursor**Type: String Default: 'auto' Defines the cursor that is being shown while sorting. Code examplesInitialize a sortable with the cursor option specified. $( ".selector" ).sortable({ cursor: 'crosshair' });Get or set the cursor option, after init. //getter var cursor = $( ".selector" ).sortable( "option", "cursor" ); //setter $( ".selector" ).sortable( "option", "cursor", 'crosshair' );

#### **cursorAt**Type: Object Default: false Moves the sorting element or helper so the cursor always appears to drag from the same position. Coordinates can be given as a hash using a combination of one or two keys: { top, left, right, bottom }. Code examplesInitialize a sortable with the cursorAt option specified. $( ".selector" ).sortable({ cursorAt: 'top' });Get or set the cursorAt option, after init. //getter var cursorAt = $( ".selector" ).sortable( "option", "cursorAt" ); //setter $( ".selector" ).sortable( "option", "cursorAt", 'top' );

#### **delay**Type: Integer Default: 0 Time in milliseconds to define when the sorting should start. It helps preventing unwanted drags when clicking on an element. Code examplesInitialize a sortable with the delay option specified. $( ".selector" ).sortable({ delay: 500 });Get or set the delay option, after init. //getter var delay = $( ".selector" ).sortable( "option", "delay" ); //setter $( ".selector" ).sortable( "option", "delay", 500 );

#### **distance**Type: Integer Default: 1 Tolerance, in pixels, for when sorting should start. If specified, sorting will not start until after mouse is dragged beyond distance. Can be used to allow for clicks on elements within a handle. Code examplesInitialize a sortable with the distance option specified. $( ".selector" ).sortable({ distance: 30 });Get or set the distance option, after init. //getter var distance = $( ".selector" ).sortable( "option", "distance" ); //setter $( ".selector" ).sortable( "option", "distance", 30 );

#### **dropOnEmpty**Type: Boolean Default: true If false items from this sortable can't be dropped to an empty linked sortable. Code examplesInitialize a sortable with the dropOnEmpty option specified. $( ".selector" ).sortable({ dropOnEmpty: false });Get or set the dropOnEmpty option, after init. //getter var dropOnEmpty = $( ".selector" ).sortable( "option", "dropOnEmpty" ); //setter $( ".selector" ).sortable( "option", "dropOnEmpty", false );

#### **forceHelperSize**Type: Boolean Default: false If true, forces the helper to have a size. Code examplesInitialize a sortable with the forceHelperSize option specified. $( ".selector" ).sortable({ forceHelperSize: true });Get or set the forceHelperSize option, after init. //getter var forceHelperSize = $( ".selector" ).sortable( "option", "forceHelperSize" ); //setter $( ".selector" ).sortable( "option", "forceHelperSize", true );

#### **forcePlaceholderSize**Type: Boolean Default: false If true, forces the placeholder to have a size. Code examplesInitialize a sortable with the forcePlaceholderSize option specified. $( ".selector" ).sortable({ forcePlaceholderSize: true });Get or set the forcePlaceholderSize option, after init. //getter var forcePlaceholderSize = $( ".selector" ).sortable( "option", "forcePlaceholderSize" ); //setter $( ".selector" ).sortable( "option", "forcePlaceholderSize", true );

#### **grid**Type: Array Default: false Snaps the sorting element or helper to a grid, every x and y pixels. Array values: [x, y] Code examplesInitialize a sortable with the grid option specified. $( ".selector" ).sortable({ grid: [50, 20] });Get or set the grid option, after init. //getter var grid = $( ".selector" ).sortable( "option", "grid" ); //setter $( ".selector" ).sortable( "option", "grid", [50, 20] );

#### **handle**Type: Selector, Element Default: false Restricts sort start click to the specified element. Code examplesInitialize a sortable with the handle option specified. $( ".selector" ).sortable({ handle: 'h2' });Get or set the handle option, after init. //getter var handle = $( ".selector" ).sortable( "option", "handle" ); //setter $( ".selector" ).sortable( "option", "handle", 'h2' );

#### **helper**Type: String, Function Default: 'original' Allows for a helper element to be used for dragging display. The supplied function receives the event and the element being sorted, and should return a DOMElement to be used as a custom proxy helper. Possible values: 'original', 'clone' Code examplesInitialize a sortable with the helper option specified. $( ".selector" ).sortable({ helper: 'clone' });Get or set the helper option, after init. //getter var helper = $( ".selector" ).sortable( "option", "helper" ); //setter $( ".selector" ).sortable( "option", "helper", 'clone' );

#### **items**Type: Selector Default: '> \*' Specifies which items inside the element should be sortable. Code examplesInitialize a sortable with the items option specified. $( ".selector" ).sortable({ items: 'li' });Get or set the items option, after init. //getter var items = $( ".selector" ).sortable( "option", "items" ); //setter $( ".selector" ).sortable( "option", "items", 'li' );

#### **opacity**Type: Float Default: false Defines the opacity of the helper while sorting. From 0.01 to 1 Code examplesInitialize a sortable with the opacity option specified. $( ".selector" ).sortable({ opacity: 0.6 });Get or set the opacity option, after init. //getter var opacity = $( ".selector" ).sortable( "option", "opacity" ); //setter $( ".selector" ).sortable( "option", "opacity", 0.6 );

#### **placeholder**Type: String Default: false Class that gets applied to the otherwise white space. Code examplesInitialize a sortable with the placeholder option specified. $( ".selector" ).sortable({ placeholder: 'ui-state-highlight' });Get or set the placeholder option, after init. //getter var placeholder = $( ".selector" ).sortable( "option", "placeholder" ); //setter $( ".selector" ).sortable( "option", "placeholder", 'ui-state-highlight' );

#### **revert**Type: Boolean/Integer Default: false If set to true, the item will be reverted to its new DOM position with a smooth animation. Optionally, it can also be set to a number that controls the duration of the animation in ms. Code examplesInitialize a sortable with the revert option specified. $( ".selector" ).sortable({ revert: true });Get or set the revert option, after init. //getter var revert = $( ".selector" ).sortable( "option", "revert" ); //setter $( ".selector" ).sortable( "option", "revert", true );

#### **scroll**Type: Boolean Default: true If set to true, the page scrolls when coming to an edge. Code examplesInitialize a sortable with the scroll option specified. $( ".selector" ).sortable({ scroll: false });Get or set the scroll option, after init. //getter var scroll = $( ".selector" ).sortable( "option", "scroll" ); //setter $( ".selector" ).sortable( "option", "scroll", false );

#### **scrollSensitivity**Type: Integer Default: 20 Defines how near the mouse must be to an edge to start scrolling. Code examplesInitialize a sortable with the scrollSensitivity option specified. $( ".selector" ).sortable({ scrollSensitivity: 40 });Get or set the scrollSensitivity option, after init. //getter var scrollSensitivity = $( ".selector" ).sortable( "option", "scrollSensitivity" ); //setter $( ".selector" ).sortable( "option", "scrollSensitivity", 40 );

#### **scrollSpeed**Type: Integer Default: 20 The speed at which the window should scroll once the mouse pointer gets within the scrollSensitivity distance. Code examplesInitialize a sortable with the scrollSpeed option specified. $( ".selector" ).sortable({ scrollSpeed: 40 });Get or set the scrollSpeed option, after init. //getter var scrollSpeed = $( ".selector" ).sortable( "option", "scrollSpeed" ); //setter $( ".selector" ).sortable( "option", "scrollSpeed", 40 );

### **tolerance**Type: String Default: 'intersect'

* This is the way the reordering behaves during drag. Possible values: 'intersect', 'pointer'. In some setups, 'pointer' is more natural.
  + **intersect**: draggable overlaps the droppable at least 50%
  + **pointer**: mouse pointer overlaps the droppable

Code examplesInitialize a sortable with the tolerance option specified.  
$( ".selector" ).sortable({ tolerance: 'pointer' });Get or set the tolerance option, after init.  
//getter  
var tolerance = $( ".selector" ).sortable( "option", "tolerance" );  
//setter  
$( ".selector" ).sortable( "option", "tolerance", 'pointer' );

#### **zIndex**Type: Integer Default: 1000 Z-index for element/helper while being sorted. Code examplesInitialize a sortable with the zIndex option specified. $( ".selector" ).sortable({ zIndex: 5 });Get or set the zIndex option, after init. //getter var zIndex = $( ".selector" ).sortable( "option", "zIndex" ); //setter $( ".selector" ).sortable( "option", "zIndex", 5 );

## Events

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

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

#### **sort**Type: sort This event is triggered during sorting. Code examplesSupply a callback function to handle the sort event as an init option. $( ".selector" ).sortable({ sort: function(event, ui) { ... } });Bind to the sort event by type: sort. $( ".selector" ).bind( "sort", function(event, ui) { ... });

#### **change**Type: sortchange This event is triggered during sorting, but only when the DOM position has changed. Code examplesSupply a callback function to handle the change event as an init option. $( ".selector" ).sortable({ change: function(event, ui) { ... } });Bind to the change event by type: sortchange. $( ".selector" ).bind( "sortchange", function(event, ui) { ... });

#### **beforeStop**Type: sortbeforestop This event is triggered when sorting stops, but when the placeholder/helper is still available. Code examplesSupply a callback function to handle the beforeStop event as an init option. $( ".selector" ).sortable({ beforeStop: function(event, ui) { ... } });Bind to the beforeStop event by type: sortbeforestop. $( ".selector" ).bind( "sortbeforestop", function(event, ui) { ... });

#### **stop**Type: sortstop This event is triggered when sorting has stopped. Code examplesSupply a callback function to handle the stop event as an init option. $( ".selector" ).sortable({ stop: function(event, ui) { ... } });Bind to the stop event by type: sortstop. $( ".selector" ).bind( "sortstop", function(event, ui) { ... });

#### **update**Type: sortupdate This event is triggered when the user stopped sorting and the DOM position has changed. Code examplesSupply a callback function to handle the update event as an init option. $( ".selector" ).sortable({ update: function(event, ui) { ... } });Bind to the update event by type: sortupdate. $( ".selector" ).bind( "sortupdate", function(event, ui) { ... });

#### **receive**Type: sortreceive This event is triggered when a connected sortable list has received an item from another list. Code examplesSupply a callback function to handle the receive event as an init option. $( ".selector" ).sortable({ receive: function(event, ui) { ... } });Bind to the receive event by type: sortreceive. $( ".selector" ).bind( "sortreceive", function(event, ui) { ... });

#### **remove**Type: sortremove This event is triggered when a sortable item has been dragged out from the list and into another. Code examplesSupply a callback function to handle the remove event as an init option. $( ".selector" ).sortable({ remove: function(event, ui) { ... } });Bind to the remove event by type: sortremove. $( ".selector" ).bind( "sortremove", function(event, ui) { ... });

#### **over**Type: sortover This event is triggered when a sortable item is moved into a connected list. Code examplesSupply a callback function to handle the over event as an init option. $( ".selector" ).sortable({ over: function(event, ui) { ... } });Bind to the over event by type: sortover. $( ".selector" ).bind( "sortover", function(event, ui) { ... });

#### **out**Type: sortout This event is triggered when a sortable item is moved away from a connected list. Code examplesSupply a callback function to handle the out event as an init option. $( ".selector" ).sortable({ out: function(event, ui) { ... } });Bind to the out event by type: sortout. $( ".selector" ).bind( "sortout", function(event, ui) { ... });

#### **activate**Type: sortactivate This event is triggered when using connected lists, every connected list on drag start receives it. Code examplesSupply a callback function to handle the activate event as an init option. $( ".selector" ).sortable({ activate: function(event, ui) { ... } });Bind to the activate event by type: sortactivate. $( ".selector" ).bind( "sortactivate", function(event, ui) { ... });

#### **deactivate**Type: sortdeactivate This event is triggered when sorting was stopped, is propagated to all possible connected lists. Code examplesSupply a callback function to handle the deactivate event as an init option. $( ".selector" ).sortable({ deactivate: function(event, ui) { ... } });Bind to the deactivate event by type: sortdeactivate. $( ".selector" ).bind( "sortdeactivate", function(event, ui) { ... });

## Methods

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

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

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

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

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

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

### **serialize**Signature: .sortable( "serialize" , [options] ) Serializes the sortable's item id's into a form/ajax submittable string. Calling this method produces a hash that can be appended to any url to easily submit a new item order back to the server. It works by default by looking at the id of each item in the format 'setname\_number', and it spits out a hash like "setname[]=number&setname[]=number". You can also give in a option hash as second argument to custom define how the function works. The possible options are: 'key' (replaces part1[] with whatever you want), 'attribute' (test another attribute than 'id') and 'expression' (use your own regexp). If serialize returns an empty string, make sure the id attributes include an underscore. They must be in the form: "set\_number" For example, a 3 element list with id attributes foo\_1, foo\_5, foo\_2 will serialize to foo[]=1&foo[]=5&foo[]=2. You can use an underscore, equal sign or hyphen to separate the set and number. For example foo=1 or foo-1 or foo\_1 all serialize to foo[]=1.

### **toArray**Signature: .sortable( "toArray" ) Serializes the sortable's item id's into an array of string. If you have <ul id="a\_sortable"><br> <li id="hello">Hello</li><br> <li id="goodbye">Good bye</li><br> </ul> and do var result = $('#a\_sortable').sortable('toArray'); then result[0] will contain "hello" and result[1] will contain "goodbye".

### **refresh**Signature: .sortable( "refresh" ) Refresh the sortable items. Custom trigger the reloading of all sortable items, causing new items to be recognized.

### **refreshPositions**Signature: .sortable( "refreshPositions" ) Refresh the cached positions of the sortables' items. Calling this method refreshes the cached item positions of all sortables. This is usually done automatically by the script and slows down performance. Use wisely.

### **cancel**Signature: .sortable( "cancel" ) Cancels a change in the current sortable and reverts it back to how it was before the current sort started. Useful in the stop and receive callback functions. If the sortable item is not being moved from one connected sortable to another: $(this).sortable('cancel'); will cancel the change. If the sortable item is being moved from one connected sortable to another: $(ui.sender).sortable('cancel'); will cancel the change. Useful in the 'receive' callback.

## Theming

The jQuery UI Sortable 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.sortable.css stylesheet that can be modified. These classes are highlighed in bold below.

### Sample markup with jQuery UI CSS Framework classes

<ul class="**ui-sortable**">

   <li></li>

   <li class="**ui-sortable-helper**"></li>

   <li class="**ui-sortable-placeholder**"></li>

   <li></li>

</ul>

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

**<ul>**

**<li></li>**

**<li></li>**

**<li></li>**

**</ul>.**