FreddyNi / **Sortable**

Code | Pull requests | Actions | Projects | Wiki | Security | Insights | Settings

Reorderable drag-and-drop lists for modern browsers and touch devices. No jQuery or framework required.

sortablejs.github.io/sortable/

MIT license

0 stars | 3.7k forks | 0 watching | 1 Branch | 0 Tags | Activity

Public repository · Forked from SortableJS/Sortable

master | 1 Branch | 0 Tags | Go to file t | Go to file | Add file | Code

This branch is up to date with `SortableJS/Sortable:master` . | Contribute | Sync fork

owen-m1 1.15.3 — dcb8f9e · 2 weeks ago

.circleci	Fixed missing forEach indexes + Fixed CircleC...	5 years ago
.github/ISSUE_TEMPLATE	moves issue templates to the correct folder	4 years ago
entry	Next version (SortableJS#1524)	5 years ago
modular	1.15.3	2 weeks ago
plugins	docs: properly capitalize forceAutoScrollFallb...	last year
scripts	SortableJS#1514: Add conditional for rootEl	5 years ago
src	Merge pull request SortableJS#2347 from dri...	6 months ago
st	update example	4 years ago
tests	Add tests	5 years ago
.editorconfig	Add tests	5 years ago
.gitignore	Next version (SortableJS#1524)	5 years ago
.jshintrc	Next version (SortableJS#1524)	5 years ago
.testcaferc.json	Add tests	5 years ago
CONTRIBUTING.md	Update CONTRIBUTING.md	5 years ago
LICENSE	Create LICENSE	5 years ago
README.md	export expando with utils + include src with n...	6 months ago
Sortable.js	1.15.3	2 weeks ago
Sortable.min.js	1.15.3	2 weeks ago
babel.config.js	Next version (SortableJS#1524)	5 years ago
bower.json	1.8.4	5 years ago
index.html	update example	4 years ago
package-lock.json	1.15.1	10 months ago
package.json	1.15.3	2 weeks ago

README | License

Sortable

Sortable is a JavaScript library for reorderable drag-and-drop lists.

Demo: http://sortablejs.github.io/Sortable/

Features

- Supports touch devices and modern browsers (including IE9)
- Can drag from one list to another or within the same list
- CSS animation when moving items
- Supports drag handles *and selectable text* (better than voidberg's html5sortable)
- Smart auto-scrolling
- Advanced swap detection
- Smooth animations
- Multi-drag support
- Support for CSS transforms
- Built using native HTML5 drag and drop API
- Supports
  - Meteor
  - Angular
    - 2.0+
    - 1.*
  - React
    - ES2015+
    - Mixin
  - Knockout
  - Polymer
  - Vue
  - Ember
- Supports any CSS library, e.g. Bootstrap
- Simple API
- Support for plugins
- CDN
- No jQuery required (but there is support)
- Typescript definitions at `@types/sortablejs`

Articles

- Dragging Multiple Items in Sortable (April 26, 2019)
- Swap Thresholds and Direction (December 2, 2018)
- Sortable v1.0 — New capabilities (December 22, 2014)
- Sorting with the help of HTML5 Drag'n'Drop API (December 23, 2013)

Getting Started

Install with NPM:

```
npm install sortablejs --save
```

Install with Bower:

```
bower install --save sortablejs
```

Import into your project:

```
// Default SortableJS
import Sortable from 'sortablejs';

// Core SortableJS (without default plugins)
import Sortable from 'sortablejs/modular/sortable.core.esm.js';

// Complete SortableJS (with all plugins)
import Sortable from 'sortablejs/modular/sortable.complete.esm.js';
```

Cherrypick plugins:

```
// Cherrypick extra plugins
import Sortable, { MultiDrag, Swap } from 'sortablejs';

Sortable.mount(new MultiDrag(), new Swap());


// Cherrypick default plugins
import Sortable, { AutoScroll } from 'sortablejs/modular/sortable.core.esm.js';

Sortable.mount(new AutoScroll());
```

Usage

```
<ul id="items">
        <li>item 1</li>
        <li>item 2</li>
        <li>item 3</li>
</ul>
```

```
var el = document.getElementById('items');
var sortable = Sortable.create(el);
```

You can use any element for the list and its elements, not just `ul` / `li`. Here is an example with `div`s.

Options

```
var sortable = new Sortable(el, {
        group: "name",  // or { name: "...", pull: [true, false, 'clone', array], put: [true, false, array] }
        sort: true,  // sorting inside list
        delay: 0, // time in milliseconds to define when the sorting should start
        delayOnTouchOnly: false, // only delay if user is using touch
        touchStartThreshold: 0, // px, how many pixels the point should move before cancelling a delayed drag event
        disabled: false, // Disables the sortable if set to true.
        store: null,  // @see Store
        animation: 150,  // ms, animation speed moving items when sorting, `0` — without animation
        easing: "cubic-bezier(1, 0, 0, 1)", // Easing for animation. Defaults to null. See https://easings.net/ for
        handle: ".my-handle",  // Drag handle selector within list items
        filter: ".ignore-elements",  // Selectors that do not lead to dragging (String or Function)
        preventOnFilter: true, // Call `event.preventDefault()` when triggered `filter`
        draggable: ".item",  // Specifies which items inside the element should be draggable

        dataIdAttr: 'data-id', // HTML attribute that is used by the `toArray()` method

        ghostClass: "sortable-ghost",  // Class name for the drop placeholder
        chosenClass: "sortable-chosen",  // Class name for the chosen item
        dragClass: "sortable-drag",  // Class name for the dragging item

        swapThreshold: 1, // Threshold of the swap zone
        invertSwap: false, // Will always use inverted swap zone if set to true
        invertedSwapThreshold: 1, // Threshold of the inverted swap zone (will be set to swapThreshold value by defa
        direction: 'horizontal', // Direction of Sortable (will be detected automatically if not given)
```

```
forceFallback: false,  // ignore the HTML5 DnD behaviour and force the fallback to kick in

fallbackClass: "sortable-fallback",  // Class name for the cloned DOM Element when using forceFallback
fallbackOnBody: false,  // Appends the cloned DOM Element into the Document's Body
fallbackTolerance: 0, // Specify in pixels how far the mouse should move before it's considered as a drag.

dragoverBubble: false,
removeCloneOnHide: true, // Remove the clone element when it is not showing, rather than just hiding it
emptyInsertThreshold: 5, // px, distance mouse must be from empty sortable to insert drag element into it


setData: function (/** DataTransfer */dataTransfer, /** HTMLElement*/dragEl) {
        dataTransfer.setData('Text', dragEl.textContent); // `dataTransfer` object of HTML5 DragEvent
},

// Element is chosen
onChoose: function (/**Event*/evt) {
        evt.oldIndex;  // element index within parent
},

// Element is unchosen
onUnchoose: function(/**Event*/evt) {
        // same properties as onEnd
},

// Element dragging started
onStart: function (/**Event*/evt) {
        evt.oldIndex;  // element index within parent
},

// Element dragging ended
onEnd: function (/**Event*/evt) {
        var itemEl = evt.item;  // dragged HTMLElement
        evt.to;    // target list
        evt.from;  // previous list
        evt.oldIndex;  // element's old index within old parent
        evt.newIndex;  // element's new index within new parent
        evt.oldDraggableIndex; // element's old index within old parent, only counting draggable elements
        evt.newDraggableIndex; // element's new index within new parent, only counting draggable elements
        evt.clone // the clone element
        evt.pullMode;  // when item is in another sortable: `"clone"` if cloning, `true` if moving
},

// Element is dropped into the list from another list
onAdd: function (/**Event*/evt) {
        // same properties as onEnd
},

// Changed sorting within list
onUpdate: function (/**Event*/evt) {
        // same properties as onEnd
},

// Called by any change to the list (add / update / remove)
onSort: function (/**Event*/evt) {
        // same properties as onEnd
},

// Element is removed from the list into another list
onRemove: function (/**Event*/evt) {
        // same properties as onEnd
},

// Attempt to drag a filtered element
onFilter: function (/**Event*/evt) {
        var itemEl = evt.item;  // HTMLElement receiving the `mousedown|tapstart` event.
},

// Event when you move an item in the list or between lists
onMove: function (/**Event*/evt, /**Event*/originalEvent) {
        // Example: https://jsbin.com/nawahef/edit?js,output
        evt.dragged; // dragged HTMLElement
        evt.draggedRect; // DOMRect {left, top, right, bottom}
        evt.related; // HTMLElement on which have guided
        evt.relatedRect; // DOMRect
        evt.willInsertAfter; // Boolean that is true if Sortable will insert drag element after target by de
        originalEvent.clientY; // mouse position
        // return false; — for cancel
        // return -1; — insert before target
```

```
                // return 1; - insert after target
                // return true; - keep default insertion point based on the direction
                // return void; - keep default insertion point based on the direction
        },

        // Called when creating a clone of element
        onClone: function (/**Event*/evt) {
                var origEl = evt.item;
                var cloneEl = evt.clone;
        },

        // Called when dragging element changes position
        onChange: function(/**Event*/evt) {
                evt.newIndex // most likely why this event is used is to get the dragging element's current index
                // same properties as onEnd
        }
});
```

`group` option

To drag elements from one list into another, both lists must have the same `group` value. You can also define whether lists can give away, give and keep a copy (`clone`), and receive elements.

- name: `String` — group name
- pull: `true|false|["foo", "bar"]|'clone'|function` — ability to move from the list. `clone` — copy the item, rather than move. Or an array of group names which the elements may be put in. Defaults to `true` .
- put: `true|false|["baz", "qux"]|function` — whether elements can be added from other lists, or an array of group names from which elements can be added.
- revertClone: `boolean` — revert cloned element to initial position after moving to a another list.

Demo:

- https://jsbin.com/hijetos/edit?js,output
- https://jsbin.com/nacoyah/edit?js,output — use of complex logic in the `pull` and `put`
- https://jsbin.com/bifuyab/edit?js,output — use `revertClone: true`

`sort` option

Allow sorting inside list.

Demo: https://jsbin.com/jayedig/edit?js,output

`delay` option

Time in milliseconds to define when the sorting should start. Unfortunately, due to browser restrictions, delaying is not possible on IE or Edge with native drag & drop.

Demo: https://jsbin.com/zosiwah/edit?js,output

`delayOnTouchOnly` option

Whether or not the delay should be applied only if the user is using touch (eg. on a mobile device). No delay will be applied in any other case. Defaults to `false` .

`swapThreshold` option

Percentage of the target that the swap zone will take up, as a float between `0` and `1` .

Read more

Demo: http://sortablejs.github.io/Sortable#thresholds

`invertSwap` option

Set to `true` to set the swap zone to the sides of the target, for the effect of sorting "in between" items.

Read more

Demo: http://sortablejs.github.io/Sortable#thresholds

`invertedSwapThreshold` option

Percentage of the target that the inverted swap zone will take up, as a float between `0` and `1`. If not given, will default to `swapThreshold`.

Read more

`direction` option

Direction that the Sortable should sort in. Can be set to `'vertical'`, `'horizontal'`, or a function, which will be called whenever a target is dragged over. Must return `'vertical'` or `'horizontal'`.

Read more

Example of direction detection for vertical list that includes full column and half column elements:

```
Sortable.create(el, {
        direction: function(evt, target, dragEl) {
                if (target !== null && target.className.includes('half-column') && dragEl.className.includes('half-c
                        return 'horizontal';
                }
                return 'vertical';
        }
});
```

`touchStartThreshold` option

This option is similar to `fallbackTolerance` option.

When the `delay` option is set, some phones with very sensitive touch displays like the Samsung Galaxy S8 will fire unwanted touchmove events even when your finger is not moving, resulting in the sort not triggering.

This option sets the minimum pointer movement that must occur before the delayed sorting is cancelled.

Values between 3 to 5 are good.

`disabled` options

Disables the sortable if set to `true`.

Demo: https://jsbin.com/sewokud/edit?js,output

```
var sortable = Sortable.create(list);

document.getElementById("switcher").onclick = function () {
        var state = sortable.option("disabled"); // get

        sortable.option("disabled", !state); // set
};
```

`handle` option

To make list items draggable, Sortable disables text selection by the user. That's not always desirable. To allow text selection, define a drag handler, which is an area of every list element that allows it to be dragged around.

Demo: https://jsbin.com/numakuh/edit?html,js,output

```
Sortable.create(el, {
        handle: ".my-handle"
});
```

```
<ul>
        <li><span class="my-handle">::</span> list item text one
        <li><span class="my-handle">::</span> list item text two
</ul>
```

```
.my-handle {
        cursor: move;
        cursor: -webkit-grabbing;
}
```

`filter` option

```
Sortable.create(list, {
        filter: ".js-remove, .js-edit",
        onFilter: function (evt) {
                var item = evt.item,
                        ctrl = evt.target;

                if (Sortable.utils.is(ctrl, ".js-remove")) {  // Click on remove button
                        item.parentNode.removeChild(item); // remove sortable item
                }
                else if (Sortable.utils.is(ctrl, ".js-edit")) {  // Click on edit link
                        // ...
                }
        }
})
```

`ghostClass` option

Class name for the drop placeholder (default `sortable-ghost`).

Demo: https://jsbin.com/henuyiw/edit?css,js,output

```
.ghost {
  opacity: 0.4;
}
```

```
Sortable.create(list, {
  ghostClass: "ghost"
});
```

`chosenClass` option

Class name for the chosen item (default `sortable-chosen`).

Demo: https://jsbin.com/hoqufox/edit?css,js,output

```
.chosen {
  color: #fff;
  background-color: #c00;
}
```

```
Sortable.create(list, {
  delay: 500,
  chosenClass: "chosen"
});
```

`forceFallback` option

If set to `true`, the Fallback for non HTML5 Browser will be used, even if we are using an HTML5 Browser. This gives us the possibility to test the behaviour for older Browsers even in newer Browser, or make the Drag 'n Drop feel more consistent between Desktop , Mobile and old Browsers.

On top of that, the Fallback always generates a copy of that DOM Element and appends the class `fallbackClass` defined in the options. This behaviour controls the look of this 'dragged' Element.

Demo: https://jsbin.com/sibiput/edit?html,css,js,output

`fallbackTolerance` option

Emulates the native drag threshold. Specify in pixels how far the mouse should move before it's considered as a drag. Useful if the items are also clickable like in a list of links.

When the user clicks inside a sortable element, it's not uncommon for your hand to move a little between the time you press and the time you release. Dragging only starts if you move the pointer past a certain tolerance, so that you don't accidentally start dragging every time you click.

3 to 5 are probably good values.

`dragoverBubble` option

If set to `true`, the dragover event will bubble to parent sortables. Works on both fallback and native dragover event. By default, it is false, but Sortable will only stop bubbling the event once the element has been inserted into a parent Sortable, or *can* be inserted into a parent Sortable, but isn't at that specific time (due to animation, etc).

Since 1.8.0, you will probably want to leave this option as false. Before 1.8.0, it may need to be `true` for nested sortables to work.

`removeCloneOnHide` option

If set to `false`, the clone is hidden by having it's CSS `display` property set to `none`. By default, this option is `true`, meaning Sortable will remove the cloned element from the DOM when it is supposed to be hidden.

`emptyInsertThreshold` option

The distance (in pixels) the mouse must be from an empty sortable while dragging for the drag element to be inserted into that sortable. Defaults to `5`. Set to `0` to disable this feature.

Demo: https://jsbin.com/becavoj/edit?js,output

An alternative to this option would be to set a padding on your list when it is empty.

For example:

```
ul:empty {
  padding-bottom: 20px;
}
```

Warning: For `:empty` to work, it must have no node inside (even text one).

Demo: https://jsbin.com/yunakeg/edit?html,css,js,output

Event object (demo)

- to: `HTMLElement` — list, in which moved element
- from: `HTMLElement` — previous list
- item: `HTMLElement` — dragged element
- clone: `HTMLElement`
- oldIndex: `Number|undefined` — old index within parent
- newIndex: `Number|undefined` — new index within parent
- oldDraggableIndex: `Number|undefined` — old index within parent, only counting draggable elements
- newDraggableIndex: `Number|undefined` — new index within parent, only counting draggable elements
- pullMode: `String|Boolean|undefined` — Pull mode if dragging into another sortable (`"clone"`, `true`, or `false`), otherwise undefined

`move` event object

- to: `HTMLElement`
- from: `HTMLElement`
- dragged: `HTMLElement`
- draggedRect: `DOMRect`
- related: `HTMLElement` — element on which have guided
- relatedRect: `DOMRect`
- willInsertAfter: `Boolean` — `true` if will element be inserted after target (or `false` if before)

Methods

option(name: `String` [, value: *]): *

Get or set the option.

closest(el: `HTMLElement` [, selector: `String`]): `HTMLElement|null`

For each element in the set, get the first element that matches the selector by testing the element itself and traversing up through its ancestors in the DOM tree.

toArray(): `String[]`

Serializes the sortable's item `data-id` 's (`dataIdAttr` option) into an array of string.

sort(order: `String[]` , useAnimation: `Boolean`)

Sorts the elements according to the array.

```
var order = sortable.toArray();
sortable.sort(order.reverse(), true); // apply
```

save()

Save the current sorting (see store)

destroy()

Removes the sortable functionality completely.

Store

Saving and restoring of the sort.

```
<ul>
        <li data-id="1">order</li>
        <li data-id="2">save</li>
        <li data-id="3">restore</li>
</ul>
```

```
Sortable.create(el, {
        group: "localStorage-example",
        store: {
                /**
                 * Get the order of elements. Called once during initialization.
                 * @param   {Sortable}  sortable
                 * @returns {Array}
                 */
                get: function (sortable) {
                        var order = localStorage.getItem(sortable.options.group.name);
                        return order ? order.split('|') : [];
                },

                /**
                 * Save the order of elements. Called onEnd (when the item is dropped).
                 * @param {Sortable}  sortable
                 */
                set: function (sortable) {
                        var order = sortable.toArray();
                        localStorage.setItem(sortable.options.group.name, order.join('|'));
```

```
                }
        }
})
```

Bootstrap

Demo: https://jsbin.com/visimub/edit?html,js,output

```
<!-- Latest compiled and minified CSS -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.1/css/bootstrap.min.css"/>


<!-- Latest Sortable -->
<script src="http://SortableJS.github.io/Sortable/Sortable.js"></script>


<!-- Simple List -->
<ul id="simpleList" class="list-group">
        <li class="list-group-item">This is <a href="http://SortableJS.github.io/Sortable/">Sortable</a></li>
        <li class="list-group-item">It works with Bootstrap...</li>
        <li class="list-group-item">...out of the box.</li>
        <li class="list-group-item">It has support for touch devices.</li>
        <li class="list-group-item">Just drag some elements around.</li>
</ul>

<script>
    // Simple list
    Sortable.create(simpleList, { /* options */ });
</script>
```

Static methods & properties

Sortable.create(el: `HTMLElement` [, options: `Object`]): `Sortable`

Create new instance.

Sortable.active: `Sortable`

The active Sortable instance.

Sortable.dragged: `HTMLElement`

The element being dragged.

Sortable.ghost: `HTMLElement`

The ghost element.

Sortable.clone: `HTMLElement`

The clone element.

Sortable.get(element: `HTMLElement`): `Sortable`

Get the Sortable instance on an element.

Sortable.mount(plugin: `...SortablePlugin|SortablePlugin[]`)

Mounts a plugin to Sortable.

Sortable.utils

- on(el `:HTMLElement` , event `:String` , fn `:Function`) — attach an event handler function
- off(el `:HTMLElement` , event `:String` , fn `:Function`) — remove an event handler
- css(el `:HTMLElement`) `:Object` — get the values of all the CSS properties
- css(el `:HTMLElement` , prop `:String`) `:Mixed` — get the value of style properties
- css(el `:HTMLElement` , prop `:String` , value `:String`) — set one CSS properties
- css(el `:HTMLElement` , props `:Object`) — set more CSS properties
- find(ctx `:HTMLElement` , tagName `:String` [, iterator `:Function`]) `:Array` — get elements by tag name
- bind(ctx `:Mixed` , fn `:Function`) `:Function` — Takes a function and returns a new one that will always have a particular context
- is(el `:HTMLElement` , selector `:String`) `:Boolean` — check the current matched set of elements against a selector
- closest(el `:HTMLElement` , selector `:String` [, ctx `:HTMLElement`]) `:HTMLElement|Null` — for each element in the set, get the first element that matches the selector by testing the element itself and traversing up through its ancestors in the DOM tree
- clone(el `:HTMLElement`) `:HTMLElement` — create a deep copy of the set of matched elements
- toggleClass(el `:HTMLElement` , name `:String` , state `:Boolean`) — add or remove one classes from each element
- detectDirection(el `:HTMLElement`) `:String` — automatically detect the direction of the element as either `'vertical'` or `'horizontal'`
- index(el `:HTMLElement` , selector `:String`) `:Number` — index of the element within its parent for a selected set of elements
- getChild(el `:HTMLElement` , childNum `:Number` , options `:Object` , includeDragEl `:Boolean`): `HTMLElement` — get the draggable element at a given index of draggable elements within a Sortable instance
- expando `:String` — expando property name for internal use, sortableListElement[expando] returns the Sortable instance of that elemenet

Plugins

Extra Plugins (included in complete versions)

- MultiDrag
- Swap

Default Plugins (included in default versions)

- AutoScroll
- OnSpill

CDN

```
<!-- jsDelivr :: Sortable :: Latest (https://www.jsdelivr.com/package/npm/sortablejs) -->
<script src="https://cdn.jsdelivr.net/npm/sortablejs@latest/Sortable.min.js"></script>
```

Contributing (Issue/PR)

Please, read this.

Contributors

Code Contributors