Dragula

Drag and drop so simple it hurts

Join us on Slack

Move stuff between these two containers. Note how the stuff gets inserted near the mouse pointer? Great stuff.

You can move these elements between these two containers

Moving them anywhere else isn't quite possible

Anything can be moved around. That includes images, **links**, or any other nested elements.

(You can still click on links, as usual!)

There's also the possibility of moving elements around in the same container, changing their position

This is the default use case. You only need to specify the containers you want to use

More interactive use cases lie ahead

Moving `<input/>` elements works just fine. You can still focus them, too.

See?

Make sure to check out the **documentation on GitHub!**

```
dragula([document.getElementById(left), document.getElementById(right)]);
```

There are plenty of events along the lifetime of a drag event. Check out **all of them** in the docs!

As soon as you start dragging an element, a `drag` event is fired

Whenever an element is cloned because `copy: true`, a `cloned` event fires

The `shadow` event fires whenever the placeholder showing where an element would be dropped is moved to a different container or position

A `drop` event is fired whenever an element is dropped anywhere other than its origin *(where it was initially dragged from)*

If the element gets removed from the DOM as a result of dropping outside of any containers, a `remove` event gets fired

A `cancel` event fires when an element would be dropped onto an invalid target, but retains its original placement instead

The `over` event fires when you drag something over a container, and `out` fires when you drag it away from the container

Lastly, a `dragend` event is fired whenever a drag operation ends, regardless of whether it ends in a cancellation, removal, or drop

```
dragula([document.getElementById(left), document.getElementById(right)])
  .on('drag', function (el) {
    el.className = el.className.replace('ex-moved', '');
  }).on('drop', function (el) {
    el.className += ' ex-moved';
  }).on('over', function (el, container) {
    container.className += ' ex-over';
  }).on('out', function (el, container) {
    container.className = container.className.replace('ex-over', '');
  });
```

Need to be able to quickly delete stuff when it spills out of the chosen containers? Note how you can easily sort the items in any containers by just dragging and dropping.

Anxious Cab Driver	Banana Boat
Thriving Venture	Orange Juice
Such **a good blog**	Cuban Cigar
Calm Clam	Terrible Comedian

```
dragula([document.getElementById(single)], {
  removeOnSpill: true
});
```

By default, dropping an element outside of any known containers will keep the element in the last place it went over. You can make elements go back to origin if they're dropped outside of known containers, too.

Moving items between containers works as usual

If you try to drop an item outside of any containers, though, it'll retain its original position

When that happens, a `cancel` event will be raised

Note that the dragged element will go back to the place you originally dragged it from, even if you move it over other containers

This is useful if you want to ensure drop events only happen when the user intends for them to happen explicitly, avoiding surprises

```
dragula([document.getElementById(left), document.getElementById(right)], {
  revertOnSpill: true
});
```

Copying stuff is common too, so we made it easy for you.

When elements are copyable, they can't be sorted in their origin container

Copying prevents original elements from being dragged. A copy gets created and *that* gets dragged instead

Whenever that happens, a `cloned` event is raised

Note that the clones get destroyed if they're not dropped into another container

You'll be dragging a copy, so when they're dropped into another container you'll see the duplication.

```
dragula([document.getElementById(left), document.getElementById(right)], {
  copy: true
});
```

Copying stuff from only one of the containers and sorting on the other one? No problem!

When elements are copyable, they can't be sorted in their origin container

Copying prevents original elements from being dragged. A copy gets created and *that* gets dragged instead

Whenever that happens, a `cloned` event is raised

Note that the clones get destroyed if they're not dropped into another container

You'll be dragging a copy, so when they're dropped into another container you'll see the duplication.

```
dragula([document.getElementById(left), document.getElementById(right)], {
  copy: function (el, source) {
    return source === document.getElementById(left)
  },
  accepts: function (el, target) {
    return target !== document.getElementById(left)
  }
});
```

Drag handles float your cruise?

+ Move me, but you can use the plus sign to drag me around.

+ Note that `handle` element in the `moves` handler is just the original event target.

+ This might also be useful if you want multiple children of an element to be able to trigger a drag event.

+ You can also use the `moves` option to determine whether an element can be dragged at all from a container, *drag handle or not*.

```
dragula([document.getElementById(left), document.getElementById(right)], {
  moves: function (el, container, handle) {
    return handle.classList.contains('handle');
  }
});
```

There are a few similar mechanisms to determine whether an element can be dragged from a certain container (`moves`), whether an element can be dropped into a certain container at a certain position (`accepts`), and whether an element is able to originate a drag event (`invalid`).

Click or Drag! Fires a click when the mouse button is released before a `mousemove` event, otherwise a drag event is fired. No extra configuration is necessary.

Clicking on these elements triggers a regular `click` event you can listen to.

Try dragging or clicking on this element.

Note how you can click normally?

Drags don't trigger click events.

Clicks don't end up in a drag, either.

This is useful if you have elements that can be both clicked or dragged.

```
dragula([document.getElementById(container)]);
```


Get it on GitHub! bevacqua/dragula