Drag-and-Drop (ii)

If you completed the last exercise successfully, you should have a web-page which includes a `<div>` that can be dragged and dropped.

The aim of this exercise is to add further `<div>` elements to the page, and modify the code so that ***any*** of them can be dragged and dropped.

In order to do this, you will need to determine which `<div>` the mouse pointer was over when the button was pressed down. Each time a mouse-down or other event occurs, an `event` object is generated. This object includes a property called `target` which is a reference to the HTML element the pointer was over when the event occurred. You can store this reference in a global variable, and then use it to indicate which element should be moved in response to mouse-movements.

You may find this exercise easier if you tackle it in stages:

- Add at least one more `<div>` to your web-page. There is no need to give it an `id` but it should have similar style properties to the existing `<div>`.

- Modify the `setupEvents()` function in your existing code so that it attaches `mousedown` event-listeners to *all* the `<div>` elements on the page. One way to do this is to obtain a list of `<div>` elements using `document.getElementsByTagName()` and then use a for loop to attach an event-listener to each `<div>` element in the list.

- Create a global variable that will hold a reference to whichever `<div>` is currently selected. It should initially be set to `null`, e.g.:

```
var selectedDiv = null;
```

- Modify the `enableDragging()` function so that it:

    - receives a reference to an `event` object as a parameter, e.g., `enableDragging(evt)` (you may already have done this as part of the previous exercise).

    - obtains the `target` property of the `event` object and stores it in the global variable you have created. e.g.:

```
selectedDiv = evt.target
```

- Modify the `dragImage()` function so that, instead of setting the position of a specified `<div>`, it sets the position of whichever `<div>` is referenced in the global variable, e.g.:

```
selectedDiv.style.left = parseInt(evt.clientX) + 'px'
```

Test your code and check that you can drag-and-drop any `<div>` on the page.

When you have dragging working correctly, extend your code so that you can drag each `<div>` from any point within its borders, not just the top-left corner. You may already have done this for a single `<div>` in the last exercise, in which case this will require only a minor change to the existing code.

If time allows, further modify your code so that the `<div>` which is being dragged always appears OVER any other elements on the page.

Each element has a Z-index which determines its position on the *Z* axis, i.e., whether it appears 'in front of' or 'behind' other elements. If you drag an element which has a low Z-index across another element which has a higher Z-index, the element you are dragging will appear to pass 'behind' the other element.

You can avoid this by doing the following at the start of each drag operation:

- set the Z-index of ALL the `<div>` elements to a low value (e.g., 0)
    - you can do this (e.g.) using a `for` loop to iterate through the array of `<div>` elements
- set the Z-index of the selected `<div>` to a high value (e.g., 1).

You can access the Z-index style property of an element in the following way:

```
<element>.style.zIndex
```

e.g.:

```
selectedDiv.style.zIndex = 1;
```