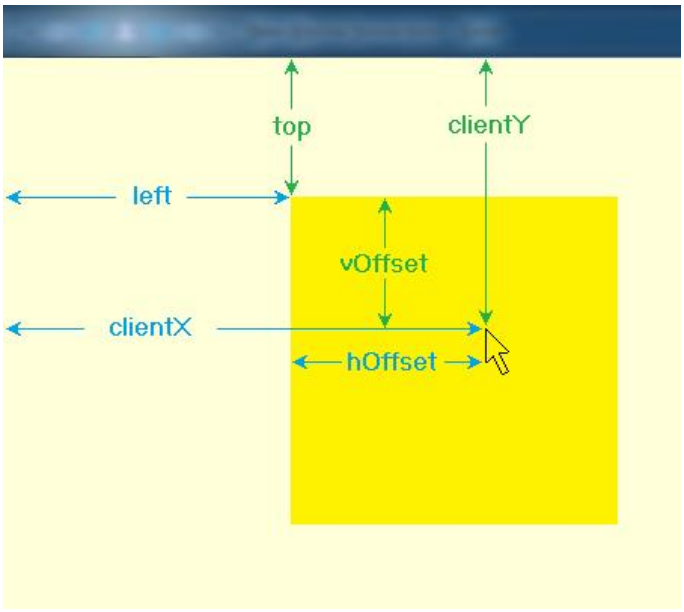
Drag-and-Drop (ii)

Drag-and-Drop with Offset

One problem with the script described in the last exercise is that the is dragged by its top-left corner.

Ideally, the drag should be centred around the cursor position at the start of the operation.



This can be corrected as follows.

At the start of the drag, calculate:

- the *horizontal offset*: `hOffset = clientX - left`
- the *vertical offset*: `vOffset = clientY - top`

... and store these values as global variables.

Each time the cursor moves, set the element's position as follows:

- `left = clientX - hOffset`
- `top = clientY - vOffset`

To implement this, first set up the required global variables, e.g.:

```
var hOffset, vOffset = 0;
```

Then modify the `startDrag()` and `dragDiv()` functions:

```
function startDrag(evt) {

  document.addEventListener('mousemove', dragDiv, false);
  document.addEventListener('mouseup', stopDrag, false);

  hOffset = parseInt(evt.clientX) - parseInt(myDiv.style.left);
  vOffset = parseInt(evt.clientY) - parseInt(myDiv.style.top);
  return false;
}
```

```
function dragDiv(evt) {

  myDiv.style.left = (parseInt(evt.clientX) - hOffset) + 'px';
  myDiv.style.top = (parseInt(evt.clientY) - vOffset) + 'px';
  return false;
}
```

Drag-and-Drop Multiple Elements

You should now have a web-page which includes a `<div>` that can be dragged and dropped.

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

First, 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>`.

```
<!DOCTYPE html>
<html>
<head>
<script type ='text/javascript'>

</script>
</head>
<body>

<div style='position:absolute; left:50px; top:50px; width:50px;
            height:50px; background-color:red'></div>

<div style='position:absolute; left:150px; top:50px; width:50px;
            height:50px; background-color:blue'></div>
</body>
</html>
```

Next, 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 *collection* of `<div>` elements using the `document.getElementsByTagName()` method
- use a `for()` loop to attach an event-listener to each `<div>` element in the list.

For example:

```
window.addEventListener('DOMContentLoaded', setupEvents, false);

function setupEvents() {

  var allDivs = document.getElementsByTagName('div');

  for (var i = 0; i < allDivs.length; i++) {

    allDivs[i].addEventListener('mousedown', startDrag, false);
  }
}
```

Now, clicking on any of the `<div>` elements will call the `startDrag()` function.

Now we need to determine which `<div>` the cursor was over when the button was pressed down. Each time a mousedown 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.

We can store this reference in a global variable, and then use it to indicate which element should be moved in response to mouse-movements.

The global variable declaration at the top of the script will now be used to hold a reference to whichever `<div>` is currently selected, so change its name to reflect this, e.g.:

```
var myDiv = null;
var selectedDiv = null;
```

Modify the `startDrag()` function so that it:

- receives a reference to an `event` object as a parameter
- obtains the `target` property of the `event` object and stores it in the global variable you have created

For example:

```
function startDrag(evt) {

  document.addEventListener('mousemove', dragDiv, false);
  document.addEventListener('mouseup', stopDrag, false);
  selectedDiv = evt.target;

  return false;
}
```

Modify the `dragDiv()` 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.:

```
function dragDiv(evt) {

  selectedDiv.style.left = parseInt(evt.clientX) + 'px';
  selectedDiv.style.top = parseInt(evt.clientY) + 'px';

  return false;
}
```

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.

This can be done in much the same way as for a single `<div>` - it requires only minor changes to the existing code.

Drag-and-Drop with Z-Index Adjustment

Using the code as described, you may find that the `<div>` which is being dragged sometimes appears to pass 'behind' the other `<div>`.

Each element on a webpage has a *Z-index* which determines its position on the *Z* axis, i.e., whether it appears 'in front of' or 'behind' other elements.

By default, each element's Z-index is determined by its position within the HTML code - the first element within the `<body>` will have the lowest Z-index, and the last element will have the highest Z-index.

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 in the `startDrag()` function:

- set the Z-index of ALL the `<div>` elements to a low value (e.g., 0)
- set the Z-index of the selected `<div>` to a higher value (e.g., 1).

For example:

```
function startDrag(evt) {

  document.addEventListener('mousemove', dragDiv, false);
  document.addEventListener('mouseup', stopDrag, false);

  for (var i = 0; i < allDivs.length; i++) allDivs[i].style.zIndex = 0;

  selectedDiv = evt.target;
  selectedDiv.style.zIndex = 1;

  return false;
}
```

Note that this uses the collection of `<div>` elements, `allDivs`, which was created in the `setupEvents()` function. Therefore, for this to work, `allDivs` must be declared as a global variable.