Drag-and-Drop (i)

The direct-manipulation style of interaction used in GUIs relies heavily on drag-and-drop, but for a long time drag-and-drop was not supported directly by web-browsers. Drag-and-drop could only be implemented using plug-ins and proprietary technologies (such as Flash).

The W3C's DOM Events Specification standardised a wide range of events within browsers, including mouse-movements, and most browsers fully support it.

Using the information below, create a web-page that uses DOM events to allow an element to be dragged-and-dropped.

1. Create a web-page that includes a `<div>` element.
   - Give the `<div>` an `id`
   - Set its `position` attribute to `absolute`, and set its `left` and `top` attributes to suitable values.
   - Set some of its style attributes to make sure that it is visible, e.g., set its `width` and `height` and give it a distinctive background colour.

2 Write a JavaScript function that attaches an event-listener to the `<div>`. The function:
   - should be called each time the page is loaded, e.g., by calling it from an `onload` event-handler in the `<body>` tag.
   - should obtain a reference to the `<div>` using (e.g.) `document.getElementById()`
   - should attach an event-listener to the `<div>` that responds to `mousemove` events and calls a function (named, e.g., `dragStart()`) whenever such an event occurs.

3. Write the function that will be called whenever a `mousemove` event is detected. It should:
   - receive a reference to the `event` object as a parameter, (e.g., `dragStart(evt)`)
   - obtain the `clientX` and `clientY` properties of the `event` object
   - display these within the `<div>`, e.g., by placing them within its `innerHTML` property.
   - return `false`.

4. View your web-page in a browser. When you move the cursor over the `<div>`, you should see the current cursor coordinates displayed within the `<div>`.

At this stage, the function (`dragStart()` in this example) is executed whenever the cursor is over the `<div>`. For drag-and-drop operation, the function should only be executed when the mouse-button is clicked and held down over the `<div>`. To achieve this, modify the code as follows.

5. Modify the function that sets-up the event-listener so that it calls the `dragStart()` function in response to `mousedown` events rather than `mousemove` events.

6. Modify the `dragStart()` function so that it, instead of reporting the cursor coordinates, it attaches two further event-listeners to the `<div>`.
   - One should call a function (e.g., `drag()`) in response to `mousemove` events .
   - The other should call a function (e.g., `dragEnd()`) in response to `mouseup` events .

7. Write a `drag()` function that is called by the event-listener you have just created. This function should obtain the `clientX` and `clientY` properties (as described in step 3, above), and use them to set the `left` and `top` style properties of the `<div>`. Note that browsers vary in the way they return coordinates, so it is best to remove any existing formatting using (e.g.) `parseInt()` and replace it with standardised formatting, for example:

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

   Make sure your function returns `false`.

8 Write a `dragEnd()` function that is called by the other event-listener you created in step 6. This function should remove the event-listeners attached to the `<div>` by the `dragStart()` function (but NOT the `mousedown` event-handler attached at start-up - this should remain in effect). This function should also return `false`.

9 View your web-page in a browser. You should find that moving the cursor initially has no effect, but that if you press the mouse-button down while cursor is over the `<div>`, the `<div>` will follow the cursor until the mouse-button is released.

   However, you may also find that it is difficult to move the `<div>` in some directions. The `<div>` will only respond to mouse-movements whilst the cursor is directly over it, so moving the cursor away from the `<div>` will cause it to stop moving.

The solution is to attach the `mousemove` and `mouseup` event-listeners to the `document`. If you do this, the events will be detected wherever they occur on the web-page, not just over the `<div>`.

10 Modify the `dragStart()` function so that it attaches its event-listeners to the `document` rather than the `<div>`. Similarly, modify the `dragEnd()` function so that it removes its event-listeners from the `document`. Do not change the `mousedown` event-handler that is attached at start-up - this should still be attached to the `<div>`.

11 View the modified web-page in a browser. You should now find that the `<div>` can be moved easily in any direction.

The code as described uses the cursor coordinates to set the *top-left corner* of the `<div>`. Thus, no matter where you click on the `<div>`, it will always be dragged by its top-left corner. It would be better if the `<div>` moved relative to the point at which the drag starts, so that if you click, e.g., in the centre of the `<div>`, the `<div>` remains centred around the cursor as you drag it.

To achieve this, add code which obtains the `top` and `left` coordinates of the `<div>` at the start of the dragging operation, stores these as variables, and uses them when positioning the `<div>` relative to the cursor. Note the following:

- Some browsers store the `top` and `left` coordinates as strings which include units, e.g., '400px'. In order to perform arithmetic on these values, you must first remove the suffix and convert the string to an integer. You can do this using `parseInt()`.
- Any function that needs to access `event` object properties must have a reference to the `event` object passed-in as a parameter (see step 3, above).