HTML5 Drag-and-Drop (i)

In this exercise you will explore the drag-and-drop functionality defined in the HTML5 standard.

- First, create a web-page that contains an image. When declaring the image, give it an `id` and make sure that it is *draggable*, e.g.:

  ```
  <img src='myImage.png' id='icon' draggable='true'>
  ```

- Write a JavaScript function to set-up the event-handling. It should add event-handlers to the image for `dragstart` and `dragend`, e.g.:

  ```
  document.getElementById('icon').addEventListener('dragstart', dragStrt, false);
  document.getElementById('icon').addEventListener('dragend', dragEnd, false);
  ```

  These event-handlers will execute a function called `dragStrt()` when the user starts to drag the image, and another function called `dragEnd()` when the user stops dragging the image.

  Arrange your code so that this function is called when the page is loaded (e.g., from on onload event-handler in the `<body>` tag).

- Next, create the `dragStrt()` and `dragEnd()` functions that will be called by the event-listeners. For the moment, the aim is just to check that the events are being captured correctly, so the `dragStrt()` function should change the appearance of the image in some way, e.g., by adding a border to it, and the `dragEnd()` function should restore the normal appearance of the image.

- View your page in a browser and check that it is working correctly. Clicking and dragging the image should cause a border to appear around it, and releasing the image should cause the border to disappear. However, you should find that you cannot move the image to a new position on the page: each time you release the image it will return to its starting position.

- In order to allow the image to be moved to a new position, you must define a target onto which it can be dropped. Create a `<div>` element on your web-page to serve as the target. Give it a suitable `id` and add styling so that is clearly visible (e.g., add borders and/or set the background to a distinctive colour).

- Add three more event-handlers to the function that sets up the event-handling. These should add event-handlers to the target `<div>` for the following events:

  - `dragenter`
  - `dragover`
  - `dragleave`

- Next, create the functions that will be called when these events occur. The function that is called on `dragenter` should change the appearance of the `<div>` in some way, e.g., by changing its background colour, and the function that is called on `dragleave` should restore its original appearance. Thus the appearance of the `<div>` will change whenever the image is dragged over it, indicating that a 'drop' can be made.

- The function that is called on `dragover` should prevent the default action of the browser. If you drag a file of a type that the browser recognises (e.g., an HTML file or a JPEG image) onto a browser window, the browser will display it in place of the existing document. The function that is called in response to `dragover` events should prevent this default action occurring. In order to do this it should obtain a reference to the event object, then use the `preventDefault()` method, e.g.:

```
function dragOver(evt) {

      evt.preventDefault();
}
```

- View your page in a browser and check that it is working correctly. Clicking and dragging the image should cause a border to appear around it, as before. Moving the image over the `<div>` should change the appearance (e.g., the background colour) of the `<div>`, and moving the image off the `<div>` should restore its original appearance.

- The next stage is to modify the code so that the image can be dropped onto the `<div>`.

  Using the HTML5 drag-and-drop events, elements can only be 'dropped' if they represent data. Add the following line to the function which is triggered by the `dragstart` event:

```
evt.dataTransfer.setData("Text", evt.target.id);
```

  This code specifies that the `id` of the image will be transferred as text.

- Add another event-handler to the function that sets up the events so that it attaches a `drop` event-handler to the target `<div>`.

- Write a function that is called when this event occurs. It should first prevent the browser's default action, in the same way described above. It should then specify what data is to be received in the event of a 'drop', and what to do with it. To do this, add the following code to the function:

```
var data = evt.dataTransfer.getData("Text");
evt.target.appendChild(document.getElementById(data));
```

  This code first obtains the transferred data (the `id` of the image) and stores it in the variable `data`. It then uses the `id` to locate the image using `document.getElementById()`, and appends it to the `<div>` using the `appendChild()` method. Thus the image should now appear inside the `<div>`.

- View your page in a browser and check that it is working correctly. It should now be possible to drag the image onto the `<div>` and, when the mouse-button is released, the image should stay there.

- When this is working correctly, create another target `<div>` similar to the first one and extend the initialisation function so that it attaches event-handlers to this `<div>` too. Test your code. It should now be possible to drop the image onto either `<div>`, and also to drag it from one to the other.