HTML5 Drag-and-Drop (i)

Using the previously-described method, images and other elements can be positioned *anywhere* on a page using drag-and-drop.

However, using drag-and-drop to support operations on files - copy, move, delete, etc. - is very difficult using this approach.

The HTML5 standard introduced a new method of drag-and-drop that is specifically designed to support operations on files and their contents.

In HTML5 drag-and-drop:

- One or more HTML elements are defined as *targets*.
- Once defined, a target responds to either of the following:
    - HTML elements that are `draggable` and have *a file or other data object* associated with them.
    - File-system icons representing files, folders, etc.
- When an icon is dropped, the information associated with it can be obtained by the target using methods provided by the *HTML5 File API*.

HTML5 includes new events to support drag-and-drop:

Existing Events	HTML5 Drag-and-Drop Events		
mousedown	dragstart	}	'icon' (dragged item)
mousemove	dragend	}	
mouseup			
click	dragenter	}	target/drop-zone
mouseover	dragover	}	
mouseout	dragleave	}	
etc.	drop	}	

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.:

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

  </script>
  </head>
  <body>
    <img src='myImage.png' id='iconImage' draggable='true'>
  </body>
</html>
```

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

```
var icon = null;

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

function setupEvents() {

  icon = document.getElementById('iconImage');

  icon.addEventListener('dragstart', dragStart, false);
  icon.addEventListener('dragend', dragEnd, false);
}
```

These event-handlers will execute:

- a function called `dragStart()` when the user starts to drag the image
- another function called `dragEnd()` when the user stops dragging the image.

For the moment, the aim is just to check that the events are being captured correctly, so:

- the `dragStart()` function should change the appearance of the image in some way, e.g., by adding a border to it.
- the `dragEnd()` function should restore the normal appearance of the image.

For example:

```
function dragStart() {

  icon.style.borderStyle = 'solid';
}

function dragEnd() {

  icon.style.borderStyle = 'none';
}
```

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: 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., set the background to a distinctive colour. For example:

```
<div id='targetDiv'
     style='width:200px; height:100px; background-color:yellow;
            position:absolute; left:50%; top:0%;'>
</div>
```

Extend the `setupEvents()` function so that it attaches event listeners to the target `<div>` as well as to the image.

The function should add event-handlers to the target `<div>` for the following events:

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

For example:

```
function setupEvents() {

  icon = document.getElementById('iconImage');
  icon.addEventListener('dragstart', dragStart, false);
  icon.addEventListener('dragend', dragEnd, false);

  target = document.getElementById('targetDiv');
  target.addEventListener('dragenter', dragEnter, false);
  target.addEventListener('dragover', dragOver, false);
  target.addEventListener('dragleave', dragLeave, false);
}
```

You should also create a global variable to hold a reference to the target `<div>`, e.g.:

```
var icon, target = null;
```

Next, create the functions that will be called when these events occur.

The `dragEnter()` and `dragLeave()` functions should change the appearance of the `<div>` in some way, e.g., by changing its background colour on `dragEnter()` and changing it back on `dragLeave()`.

Thus the appearance of the `<div>` will change whenever the image is dragged over it, indicating that a 'drop' can be made.

The `dragOver()` function 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.

For example:

```
function dragEnter() {

  target.style.backgroundColor = 'red';
}

function dragOver(evt) {

  evt.preventDefault();
}
```

```
function dragLeave() {

  target.style.backgroundColor = 'yellow';
}
```

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 background colour of the `<div>`…
- …and moving the image off the `<div>` should restore the original background colour.

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. Therefore, we need to attach some 'data' to the icon.

Modify the `dragStart()` function as follows:

```
function dragStart(evt) {

  icon.style.borderStyle = 'solid';
  evt.dataTransfer.setData("Text", evt.target.id);
}
```

This code:

- Obtains the `id` of the image (`evt.target.id`).
- Sets the `id` as data to be transferred when the image is dropped (`evt.dataTransfer.setData()`).

Further modify the `setupEvents()` function by adding a `drop` event listener to the target `<div>`.

For example:

```
function setupEvents() {

  icon = document.getElementById('iconImage');
  icon.addEventListener('dragstart', dragStart, false);
  icon.addEventListener('dragend', dragEnd, false);

  target = document.getElementById('targetDiv');
  target.addEventListener('dragenter', dragEnter, false);
  target.addEventListener('dragover', dragOver, false);
  target.addEventListener('dragleave', dragLeave, false);
  target.addEventListener('drop', dropIcon, false);
}
```

Write a `dropIcon()` function that will be called when this event occurs.

It should first prevent the browser's default action, in the same way as in the `dragOver()` function.

It should then specify what data is to be received in the event of a 'drop', and what to do with it.

For example:

```
function dropIcon(evt) {

  evt.preventDefault();
  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, extend the code so that there are two target `<div>` elements on the page, and you can drop the `<div>`

To do this:

- create another target `<div>` similar to the first one
- extend the `setupEvents()` 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.