HTML5 Drag-and-Drop (ii)

If you completed the last exercise successfully, you will have a web-page that includes an image which can be dragged-and-dropped onto one or more target areas.

This illustrates the use of HTML5 drag-and-drop *within* a web-page.

However, a key feature of the HTML5 standard is that it allows you to drag icons representing files, etc., onto a web-page and open them using JavaScript.

In this exercise you will create a web-page that includes a target area onto which items (text-files, images, etc.) can be dragged, from the desktop or elsewhere.

The HTML5 drag-and-drop API will be used to obtain information about the files and display this.

- First, create a web-page that includes a `<div>` element.
    - This will serve as the drop-target, so give it a suitable `id`, e.g., `dropArea`.
    - Add some styling to differentiate the `<div>` from the rest of the page, e.g. borders and/or a distinctive background colour.
- Add a `<textarea>` to serve as a message window, and give it a suitable `id`.
    - The appearance of the `<textarea>` is not important.

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

    </script>
  </head>
  <body>
    <div id='dropArea'
         style='width:100%; height:100px; border-style:solid'>
          Drop File Here
    </div>

    <textarea id='displayArea'
              style='width:100%; height:300px; border-style:solid'>
    </textarea>

  </body>
</html>
```

Write a JavaScript function to set-up the event-handling, etc. This function should:

- check that the browser supports the HTML5 File API
- obtain a reference to the `<div>`
- add event-listeners to the `<div>` for the `dragenter`, `dragover`, `dragleave` and `drop` events.

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

var target = null;

function setupEvents() {

  if(window.File && window.FileList && window.FileReader) {

    target = document.getElementById('dropArea');
    target.addEventListener('dragenter', dragEnter, false);
    target.addEventListener('dragover', dragOver, false);
    target.addEventListener('dragleave', dragLeave, false);
    target.addEventListener('drop', dropFile, false);
  }
  else alert('HTML 5 File API not supported');
}
```

Having set-up the event-handling, the next step is to create the functions.

For the moment, the purpose of these functions is merely to indicate that a 'drop' is possible, i.e., that the element which has been dragged over the `<div>` carries data. Therefore:

- the `dragEnter` function should highlight the `<div>` in some way, e.g., change its background colour
- the `dragLeave` and `dropFile` functions should remove the highlight.

The `dragOver` function doesn't need to change the appearance of `<div>` in any way, but it should cancel the default behaviour of the browser.

Most browsers allow you to open files by dragging them into the browser window, but in this case we want to receive and analyse files, not open them.

Therefore, the `dragOver` function - and all the other functions too - should cancel the default behaviour of the browser using the `stopPropagation()` and `preventDefault()` functions.

```
function dragEnter(evt) {

  evt.stopPropagation();
  evt.preventDefault();
  target.style.backgroundColor = 'red';
}

function dragLeave(evt) {

  evt.stopPropagation();
  evt.preventDefault();
  target.style.backgroundColor = 'white'
}

function dragOver(evt) {

  evt.stopPropagation();
  evt.preventDefault();
}

function dropFile(evt) {

  evt.stopPropagation();
  evt.preventDefault();
  target.style.backgroundColor = 'white'
}
```

Test your script:

- If you drag (e.g.) an image file from the desktop onto the target `<div>`, the background colour of the `<div>` should change to red...
- and when you move the image off of the `<div>` the background colour should revert to white.
- However, if you drag the image onto an empty area of the web-page, it should be displayed by the browser, replacing your HTML file.

Once this is working correctly, the next step is to modify the `dropFile()` function so that it obtains details of the dropped file and displays them.

This is done using the `files` property of the `dataTransfer` object.

The `files` property is an array of files, so if only one file has been dragged-and-dropped it will appear as element `0` in this array, e.g.:

```
var file = evt.dataTransfer.files[0];
```

You can now obtain a reference to the `<textarea>` you created earlier, then display the name of the file in the `<textarea>`.

```
function dropFile(evt) {

  evt.stopPropagation();
  evt.preventDefault();

  target.style.backgroundColor = 'white'
  var file = evt.dataTransfer.files[0];
  document.getElementById('displayArea').value = file.name;
}
```

Test your page in a browser.

You should be able to drag a file from the desktop onto the target `<div>`, whereupon the name of the file should appear in the `<textarea>`.

As shown above, the script only displays the *name* of the dropped file (`file.name`).

Extend your code so that it also displays the size of the file (`file.size`) and its type (`file.type`)

It's possible to drag more than one file at a time.

If multiple files are dragged, the `files` array will contain more than one entry.

Modify your code so that, if more than one file is dragged onto the target `<div>`, it displays the names, etc., of ALL the files.