# 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. 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.

- Write an initialisation function - it should be placed within `<script>` tags in the `<head>` of your web-page. This function should first check that the browser supports the HTML5 File API, e.g.:

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

  Arrange your function so that, if the conditional test returns `true`, it posts a 'welcome' message (using an `alert()` dialog-box), while if the test returns `false` it posts a warning message and ends.

  Add an event-handler that calls your initialisation function when the page loads, e.g., using an `onload` event-handler in the `<body>` tag.

- Test your page in a browser and make sure that you get the 'welcome' message, indicating that the necessary API is supported.

- If all is well, go back to your initialisation function and replace the 'welcome' message with code to set-up the events.

  In order to set-up the events, you must first obtain a reference to the target `<div>` you created earlier, then add the necessary event-handlers, e.g.:

  ```
  var dropTarget = document.getElementById('dropArea');
  dropTarget.addEventListener("dragover", dragOver, false);
  dropTarget.addEventListener("drop", dropFile, false);
  ```

  where `dropArea` is the `id` of the `<div>` on which files will be dropped, and `dragOver` and `dropFile` are the names of functions which will handle the events (see below).

- Having set-up the event-handling, you now need to create the `dragOver` and `dropFile` functions.

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 default action must be cancelled. This is done using the methods `stopPropagation()` and `preventDefault()`, e.g.:

```
function dragOver(evt) {

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

Create two functions called `dragOver()` and `dropFile()`, and place these lines of code in both. Don't forget to pass a reference to the event object to each function as a parameter (`evt`).

- Test the script. If you drag (e.g.) an image file from the desktop onto an empty area within the browser window, the image should be displayed by the browser, replacing the page which contains your script. However, if you drag the file onto the target `<div>` it should have no effect.

- As well as cancelling the default behaviour, the `dropFile()` function should obtain details of the dropped file and display 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. Therefore you can access the file by adding the following code to your `dropFile()` function:

```
var file = evt.dataTransfer.files.item(0);
```

or:

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

Add further code to your `dropFile()` function so that it obtains a reference to the `<textarea>` you created earlier (e.g., by giving the text-area an `id` and using `document.getElementById()`), then display the `name` of the file in the `<textarea>`,:

```
document.getElementById('myTextArea').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>`.

When this is working, extend your code as follows:

- In the initialisation routine, attach a `dragleave` event-handler to the target `<div>` along with the other event-handlers.
  - The existing `dragover` event will be triggered when a file is dragged over the `<div>`.
  - The `dragleave` event-handler will be triggered if the file is dragged over the `<div>`, then dragged off it without being dropped.

Modify the function that is called by the `dragover` event-handler so that it highlights the `<div>` in some (e.g., changes the background colour of the `<div>`). Create a new function that is called by the `dragleave` event-handler, and arrange the code so that it returns the `<div>` to normal when the file is moved away again.

- Test your code: dragging an icon over the `<div>` should highlight it, and dragging the icon away again should remove the highlighting.

- 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.