HTML5 Drag-and-Drop (iii)

We have seen how HTML5 drag-and-drop can be used to access the name, etc., of a file dragged from the desktop onto a web-page.

Another feature of the HTML5 drag-and-drop API is the `FileReader` object. It allows various features of a dropped file to be accessed, including the *contents* of the file. In this exercise you will explore some of the features of this object.

- Create a web-page that includes a `<div>`. This will serve as the target onto which files will be dropped. Give it an `id`, and add some styling so that it is distinguishable from the background (e.g., borders and/or a background-colour).
- Create an initialisation function that runs when the page is fully-loaded (e.g., called by an `onload` event in the `<body>` tag).
  - This function should first check that the browser supports the HTML5 File API, as in the previous exercise.
  - It should then add event-handlers to the `<div>` for the following events:

    ```
    dragenter
    dragover
    dragleave
    drop
    ```

    Arrange the event-handlers so that each calls a different function.
- Having set-up the event-handling, you now need to create the necessary functions. As in the previous exercise, it's essential to cancel the default action of the browser, so each of the functions should call the `stopPropagation()` and `preventDefault()` methods.
- Edit the function that is called when a `dragenter` event occurs so that it highlights the `<div>` in some way (e.g., changes its background-colour), and similarly edit the function that is called when a `dragleave` event occurs so that it removes the highlighting.

  Test your page in a browser. Dragging a file from (e.g.) the desktop over the `<div>` should cause the highlight to appear, and moving the file away again should cause the highlight to disappear.

- Add an `<img>` tag to your web-page. Give it an `id`, but leave the `src` attribute empty - there's no need to specify an image because this will be added later by a script.
- Add the following code to the function that is called when a `drop` event occurs:

  ```
  var files = evt.dataTransfer.files;
  if (files.length > 0) handleFiles(files);
  ```

  This code:

  - Obtains an array of the files (if any) that have been dropped by the `dataTransfer` object, and stores it in the variable `files`

  - Checks to see if any files have been dropped by noting the length of the `files` array. If `files.length` is greater than zero, it calls a function `handleFiles()` and passes the file-array to it as a parameter.

- Next create the function `handleFiles()` that is called by the code above. It should contain the following code:

  ```
  var reader = new FileReader();
  reader.onload = handleReaderLoad;
  reader.readAsDataURL(files[0]);
  ```

  This code:

  - Creates a new instance of the `FileReader()` object and stores a reference to it in the variable `reader`
  - Creates an association such that, when an `onload` event occurs on the `FileReader()` object, the function `handleReaderLoad` is called
  - Specifies that the `FileReader()` object should obtain the URL of the first (or only) file in the file-array (`files[0]`).

- Finally, create the function `handleReaderLoad` that is called by the code above. It should obtain a reference to the `<img>` tag, then set the `src` attribute of the image equal to `evt.target.result`.

Test your page in a browser. Dragging an image file from (e.g.) the desktop onto the `<div>` should cause the dragged image to appear in the `<img>` tag.

In this example, the *URL* of the image-file is being passed during a drag-and-drop operation. It is also possible to transfer the *contents* of a file.

Make a copy of your web-page, then modify the code as follows:

- In the `handleFiles()` function, change

  ```
  reader.readAsDataURL(files[0]);
  ```

  to

  ```
  reader.readAsText(files[0]);
  ```

- Replace the `<img>` tag with a `<textarea>` tag.

- In the `handleReaderLoad()` function, change

  ```
  image.src = evt.target.result;
  ```

  to

  ```
  myTextArea.value = evt.target.result;
  ```

  (where `myTextArea` is the `id` of your `<textarea>`.

Test your page in a browser. Dragging a text file from (e.g.) the desktop onto the `<div>` should cause the text in the file to appear in the `<textarea>`.