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).

```
<!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>
</body>
</html>
```

- Create a `setupEvents()` function that runs when the page is fully loaded (e.g., using an event-listener which fires when the `DOMContentLoaded` event occurs).
  - 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.

```
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, 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.
  - Arrange 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).
  - Similarly, arrange the functions that are called when the `dragleave` and `drop` events occurs so that they remove the highlighting.

```
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 page in a browser.
  - Dragging a file from (e.g.) the desktop over the `<div>` should cause the highlight to appear.
  - Moving the file away again, or dropping it, 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.

```
<!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>
  <img id='imageHolder' src='' style='border-style:solid'>
</body>
</html>
```

- Add the following code to the function that is called when a `drop` event occurs:

```
function dropFile(evt) {

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

  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:

```
function handleFiles(fileArray) {

  var reader = new FileReader();
  reader.addEventListener('load', handleReaderLoad, false);
  reader.readAsDataURL(fileArray[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 a `load` event occurs on the `FileReader()` object, a function `handleReaderLoad` is called
  - Specifies that the `FileReader()` object should obtain the URL of the first (or only) file in the file-array (`fileArray[0]`).

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

  For example:

```
function handleReaderLoad(evt) {

  var imgTag = document.getElementById('imageHolder');
  imgTag.src = 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, only the *URL* of the image-file is being passed during a drag-and-drop operation, but 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(fileArray[0]);`
- to
  - `reader.readAsText(fileArray[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>`.