Let's go further and also add an `<input type="file">`

This time we will create an example that allows files to be selected with a file chooser or by drag and dropping them, like in the screenshot below (the interactive example is a bit further down the page):

In the above screenshot, which is derived from the example detailed later in this page, we selected some files using the first button (which is an `<input type="file" multiple.../>`), then we used the second button (which is

an `<input type="file" webkitdirectory>`, Chrome only but fun to try here) to select a directory that contained 11 files. We then dragged and dropped some other images to the drop zone. Each time, thumbnails were displayed. Both methods (file selector or drag and drop) produced the same results.

IDEA: REUSE THE SAME CODE FOR READING IMAGE FILES AND DISPLAYING THUMBNAILS

If you look (again) at the very first example that displayed thumbnails, without drag'n'drop), you will notice that the event handler we used to track the selected files using `<input type="file"/>` looks like this:

```
<script>
   function handleFileSelect(evt) {
       var files = evt.target.files; // FileList object
       // do something with files... why not call readFilesAndDisplayPreview!
       readFilesAndDisplayPreview(files);
   }

 document.getElementById('files').addEventListener('change', handleFileSelect, false);
</script>
...
<body>
   Choose multiple files :<input type="file"id="files" multiple /><br/>
</body>
```

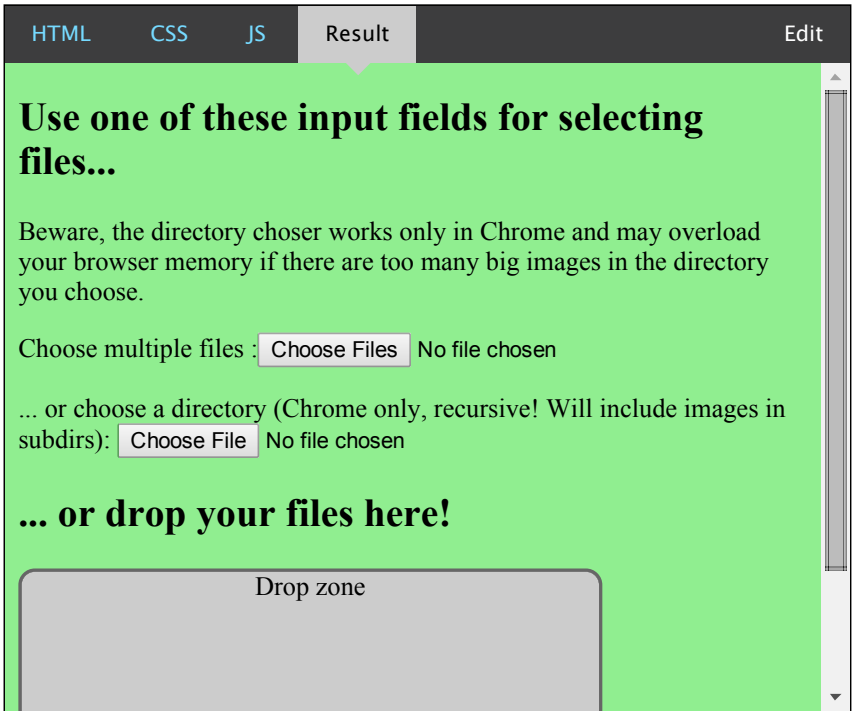
It calls `readFilesAndDisplayPreview()` at line 5! The same function with the same parameters is also used by the example that used drag'n'drop that we discussed on a previous page of this course.

Let's mix both examples: add to our drag'n'drop example an`<input type="file">` element, and the above handler. This will allow us to select files either with drag'n'drop or by using a file selector.

Just for fun, we also added an experimental "directory chooser" that is thus far only implemented by Google Chrome (notice, `<input type="file" webkitdirectory>` is not in the HTML5 specification, we added it here just for fun. The drag and drop and file chooser will work with any modern browser, but the directory chooser will only work with Google Chrome).

COMPLETE INTERACTIVE EXAMPLE WITH SOURCE CODE

Try it in your browser below (use one of the two file/dir selector or drag and drop image files in the drop zone), or play with it at CodePen:



Complete source code:

```
<!DOCTYPE html>
<html lang="en">
 <head>
   <style>
      div {
          height: 150px;
          width: 350px;
          border: 2px solid #666666;
          background-color: #ccc;
          margin-right: 5px;
          border-radius: 10px;
          box-shadow: inset 0 0 3px #000;
          text-align: center;
          cursor: move;
      }

      .dragged {
          border: 2px dashed #000;
          background-color: green;
      }
      .draggedOver {
          border: 2px dashed #000;
          background-color: green;
      }

 </style>
 <script>
      function dragLeaveHandler(event) {
```

```
30.          console.log("drag leave");
             // Set style of drop zone to default
             event.target.classList.remove('draggedOver');
          }
          function dragEnterHandler(event) {
             console.log("Drag enter");
             // Show some visual feedback
             event.target.classList.add('draggedOver');
          }
40.
          function dragOverHandler(event) {
             //console.log("Drag over a droppable zone");
             // Do not propagate the event
             event.stopPropagation();
             // Prevent default behavior, in particular when we drop
             // images or links
             event.preventDefault();
          }
          function dropHandler(event) {
51.          console.log('drop event');
             // Do not propagate the event
             event.stopPropagation();
             // Prevent default behavior, in particular when we drop
             // images or links
             event.preventDefault();
             // reset the visual look of the drop zone to default
             event.target.classList.remove('draggedOver');
62.
             // get the files from the clipboard
             var files =event.dataTransfer.files;
             var filesLen = files.length;
             var filenames = "";
             // iterate on the files, get details using the file API
             // Display file names in a list.
             for(var i = 0 ; i < filesLen ; i++){
72.             filenames += '\n' +files[i].name;
                // Create a li, set its value to a file name, add it to the ol
                var li =document.createElement('li');
                li.textContent = files[i].name;
                document.querySelector("#droppedFiles").appendChild(li);
             }
             console.log(files.length + ' file(s) have been dropped:\n' + filenames);
             readFilesAndDisplayPreview(files);
          }
83.       functionreadFilesAndDisplayPreview(files) {
             // Loop through the FileList and render image files as
             // thumbnails.
             for (var i = 0, f; f = files[i];i++) {
                // Only process image files.
                if (!f.type.match('image.*')) {
                   continue;
                }
```

```
            var reader = new FileReader();
94.
            //capture the file information.
            reader.onload = function(e) {
               // Render thumbnail.
               var span =document.createElement('span');
               span.innerHTML = "<img class='thumb' width='100' src='" +
                                e.target.result + "'/>";
               document.getElementById('list').insertBefore(span, null);
            };
           // Read the image file as a data URL.
           reader.readAsDataURL(f);
        }
     }
     function handleFileSelect(evt) {
        var files = evt.target.files; // FileList object
        // do something with files... why not call
        // readFilesAndDisplayPreview!
       readFilesAndDisplayPreview(files);
     }
114.  </script>
 </head>
<body>
 <h2>Use one of these input fields for selecting files</h2>
<p>Beware, the directory choser works only in Chrome and may overload
your browser memory if there are too many big images in the
directory you choose.</p>
Choose multiple files : <input type="file"id="files" multiple
                              onchange="handleFileSelect(event)"/>
 </p>
 <p>Choose a directory (Chrome only): <inputtype="file"
                                     id="dir" webkitdirectory
                                     onchange="handleFileSelect(event)"/>
 </p>

129.  <h2>Drop your files here!</h2>
    <div id="droppableZone"ondragenter="dragEnterHandler(event)"
                           ondrop="dropHandler(event)"
                           ondragover="dragOverHandler(event)"
                           ondragleave="dragLeaveHandler(event)">
        Drop zone
        <ol id="droppedFiles"></ol>
     </div>
 <br/>
 <output id="list"></output>
 <body>
 <html>
```

The parts that we have added are in bold. As you can see, all methods share the same code for previewing the images.