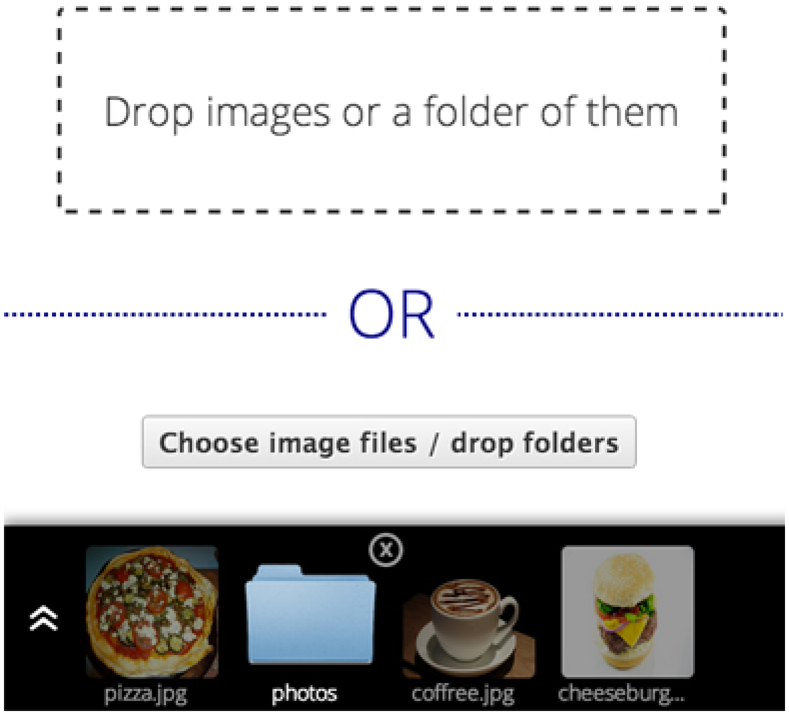
Integrating input[type=file] with the Filesystem API

By Eric Bidelman

Engineer @ Google working on web tooling: Headless Chrome, Puppeteer, Lighthouse

Let's say you have a photo editing app and you'd like users to be able to drag in hundreds of photos and copy them into your app. Ok, what do you do?


Drop images or a folder of them
OR
Choose image files / drop folders
pizza.jpg
photos
coffree.jpg
cheeseburg...

Launch Demo

In a recent post, Eiji Kitamura highlighted a subtle, yet powerful new feature in the drag and drop APIs; the ability to drag in folders *and* retrieve them as HTML5 Filesystem API `FileEntry` and `DirectoryEntry` objects (done by accessing a new method on the DataTransferItem, `.webkitGetAsEntry()`).

What's remarkably cool about the `.webkitGetAsEntry()` extension is how elegant it makes importing files and entire folders. Once you have a `FileEntry` or `DirectoryEntry` from a drop event, it's a matter of using the Filesystem API's `copyTo()` to get it imported into your app.

An example of copying multiple dropped folders over to the filesystem:

```
var fs = null; // Cache filesystem for later.

// Not shown: setup drag and drop event listeners.
function onDrop(e) {
  e.preventDefault();
  e.stopPropagation();

  var items = e.dataTransfer.items;

  for (var i = 0, item; item = items[i]; ++i) {
    var entry = item.webkitGetAsEntry();

    // Folder? Copy the DirectoryEntry over to our local filesystem.
    if (entry.isDirectory) {
      entry.copyTo(fs.root, null, function(copiedEntry) {
        // ...
      }, onError);
    }
  }
}

window.webkitRequestFileSystem(TEMPORARY, 1024 * 1204, function(fileSystem) {
  fs = fileSystem;
}, function(e) {
  console.log('Error', e);
});
```

Very nice! Again, the simplicity comes from integrating DnD with the Filesystem API calls.

Taking this one step further, we also have the ability to drag and drop a folder and/or files onto a normal `<input type="file">`, then access the entries as Filesystem directory or file entries. That is done through `.webkitEntries`:

```
<input type="file" multiple>

function onChange(e) {
  e.stopPropagation();
  e.preventDefault();

  var entries = e.target.webkitEntries; // Get all dropped items as FS API entrie

  [].forEach.call(entries, function(entry) {

    // Copy the entry into our local filesystem.
    entry.copyTo(fs.root, null, function(copiedEntry) {
      ...
    }, onError);
  });
}

document.querySelector('input[type="file"]').addEventListener('change', onChange)
```

I've put together a photo gallery demo to demonstrate these different techniques for importing files/folders.

Launch Demo

To learn more about the HTML5 Filesystem API, see Exploring the Filesystem APIs.
