

Read file content as data URL

INTRODUCTION TO A DATA URL

What is a data URL?

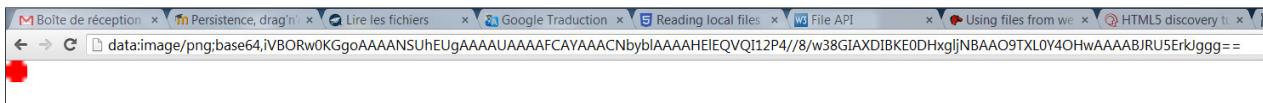
A data URL is a URL that includes type and content at the same time. It is useful, for example, for inlining images or videos in the HTML of a Web page (on mobile devices, this may speed up the loading of the page by reducing the number of HTTP requests).

Here is an example of a red square, as a data URL. Copy and paste it in the address bar of your browser, and you should see the red square:



```
data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAAUAAAFCAYAAACNbyb1AAAHE1EQVQI12P4//8/w38GIAXDIBKE0DHxgljNBAAO9TXL0Y4OHwAAAABJRU5ErkJgg==
```

This data URL in a browser address bar should look like this:



If we set the `src` attribute of an image element `` with the data URL of the above screenshot, it will work exactly as if you used a URL that started with `http://`

In your browser, you will see a small red circle rendered by this source code:

```

```

And here is the result:



This `dataURL` format enables file content to be stored in a base64 format (as a string), and adds the MIME type specification of the content. The `dataURL` can therefore store a file as a URL readable with modern browsers. It is becoming more commonly used on the Web, especially for mobile applications, as inlining images reduces the number of HTTP requests and makes the Web page load faster.

You will find lots of Web sites and tools for generating `dataURL` from files, such as [the DataURL maker Web site](#) (screenshot below):

Screenshot of the Data URL Maker tool on dataurl.net. The page has a red header with tabs: About, Data URL Maker (selected), CSS Optimizer, and Downloads.

Data URL Maker

Select or drag a file to get the Data URL: Choisissez un fichier 1902809_...3_n.jpg

1902809_10151960598335679_422509123_n.jpg

data:image/jpeg;base64,/9j/4AAQSkZJRgABAgAAAQABA
AD/7QCEUGhvG9zaG9wiDMuMAA4QkINBAQAAAAAG
ccAigAYKZCTUQwMTAwMGFhYjAzMDAwMGE1MDUjwM
DAwzWQwODAwMDBjNTA5MDAwMDdmMGEwMDAwM
TywZjAwMDA5NDE1MDAwMDBIMTYwMDAwZDUxNjAw
MDBhMTE3MDAwMDk0MjEwMDAwAP/iAhxJQ0NfUFJPR
kIMRQABAQAAgxsY21zAHAAG1udHSR0lgWFlaAfA
AEAGQADACKAOWFjic3BBUFBMAAAAAAAAAAAAAAAA
AAAAAAAAD21gABAaaaANMtbgNtcwAAAA
AAAAAAAACmRlc2MAAAD8AAAXmNwcn
QAAAFcAAAAC3d0cHQAAAFOAAAAGJrcHQAAA8AAA
AFHJYVVoAAAGQAAAAGdYVVoAAAGKAAGFGJYVW
oAAAG4AAAfhJUUKMAAHMAAAAGQGUUkMAAHM
AAAAGJUUKMAAHMAAAAGRlc2MAAAAAAAA2M
yAAAAAAAACAAAAGAAAAGAAAAGAAAAGAAA
AAAAAAAAGAAAAGAAAAGAAAAGAAAAGAAAAGAAA
AAAAAAAAGAAAAGAAAAGAAAAGAAAAGAAAAGAAA
ARKIAAFhZWiaAAAAAD21gABAaaaANMtbgNtcwAAAAA

Data URL Size: 11487 bytes
Original size: 8596 bytes

Image preview

M.Buffa playing guitar for halloween

Image preview

dataURL, notice that this encoding is larger than original jpg format.

With the above example, you can copy and paste the characters on the left and use them with an ``. Just set the `src` attribute with it!

Notice that you can encode any type of file as `dataURL`, but this format is most frequently used with media files (images, audio, video).

Example of HTML5 logo embedded in a document without any real image, just a `dataURL` and CSS

[Try it at JsBin](#)

File ▾ Add library Share HTML CSS JavaScript Console Output

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Example of data
URL</title>
</head>
<body>
  <div class="standard-HTML5">
```

insert the logo before elements with CSS class "standard-HTML5"

HTML5 logo encoded as a data URL

```
.standard-HTML5 {
  height: 63px;
  width: 57px;
}

.standard-HTML5:before {
  content:
    url(data:image/svg+xml;base64,PD94
bWwgdmVyc2lvbj0iMS4wIiBsbmNvZGlub
0iVVVRGLTgiIHN0YW5kYWxvbmU9Im5vIj8+
CjxzdmcKCXhtbG5zPSJodHRwOi8vd3d3Ln
czLm9yZy8yMDAwL3N2ZyIKCXZlcnPb249
IjEuMSIKCXZpZXdCb3g9IjAgMCA0NTegNT
EyIj4KCjx0aXRzT5IVE1MNSBMb2dvIEjh
ZGdlPC90aXRzT4KCjxwYXRoCWZpbGw9Ii
NFMzRGMjYicWQ9Ik0gNDEsNDYwIDASMC
0NTEsMC0MTAsNDYwIDIyNSw1MTIiIC8+Cj
xwYXRoCWZpbGw9IiNFRjY1MkEiCWQ9Ik0g
MjI2LDQ3MiAzNzUsNDMxDQxCwzNyAyMj
YsMzciIC8+CjxwYXRoCWZpbGw9IiNFQkVC
RUIiCWQ9Im0gMjI2LDIwOCAtNzUsMCatNS
wtNTggODAsMCawLC01NiAtMSwwICoxNDEs
MCAxLDE1IDE0LDE1NiAxMjcsMCB6IG0gMC
wxNDcgLTEsMCAtNjMsLTE3IC00LC00NSAT
MzAsMCAtMjYsMCA3LDg5IDEExNiwzMiaxLD
AgeiIgLz4KPHBhdGgJZmlsbD0iI0ZGRkZG
RiIjZD0ibSAyMjUsMjA4IDA5NTcgNzAsMC
AtNyw3MyAtNjMsMTcgMCw1OSAxMTYsLTMy
IDEsLTEwIDEzLC0xNDkgHiwtMTUgLTE2LD
AgeiBtIDAsLTExNCAwLDM1IDA5MjEgMCww
IDEzNywwIDAsMCawLDAgMSwtMTIgMywtMj
kgMSwtMTUgeiIgLz4KCjwvc3ZnPgo=);}
```



EXAMPLE 1: READ IMAGES AS DATA URL AND DISPLAY PREVIEWS IN THE PAGE

Example 1 is useful for forms that allow the user to select one or more pictures. Before sending the form, you might want to get a preview of the pictures in the HTML page. The `reader.readAsDataURL` method is used for that.

[Example on JS Bin](#) or try it below in your browser:

Choose multiple images files: No file chosen

Preview of selected images:

Source code extract:

```
<label for="files">Choose multiple files:</label>
<input type="file" id="files" multiple
  onchange="readFilesAndDisplayPreview(this.files);"/><br/>
<p>Preview of selected images:</p>
<output id="list"></output>

<script>
  function readFilesAndDisplayPreview(files) {
    // Loop through the FileList and render image files as thumbnails.
10.   for (var i = 0, f; f = files[i]; i++) {

      // Only process image files.
      if (!f.type.match('image.*')) {
```

```

        continue;
    }

    var reader = new FileReader();

    //capture the file information.
20.   reader.onload = function(e) {
        // Render thumbnail. e.target.result = the image content
        // as a data URL
        // create a span with CSS class="thumb", for nicer layout
        var span = document.createElement('span');
        // Add an img src=... in the span, with src= the dataURL of
        // the image
        span.innerHTML = "<imgclass='thumb' src='" +
            e.target.result + "' alt='a picture'/>";
        // Insert the span in the output id=list
31.   document.getElementById('list').insertBefore(span, null);
};

// Read in the image file as a data URL.
reader.readAsDataURL(f);
}
}

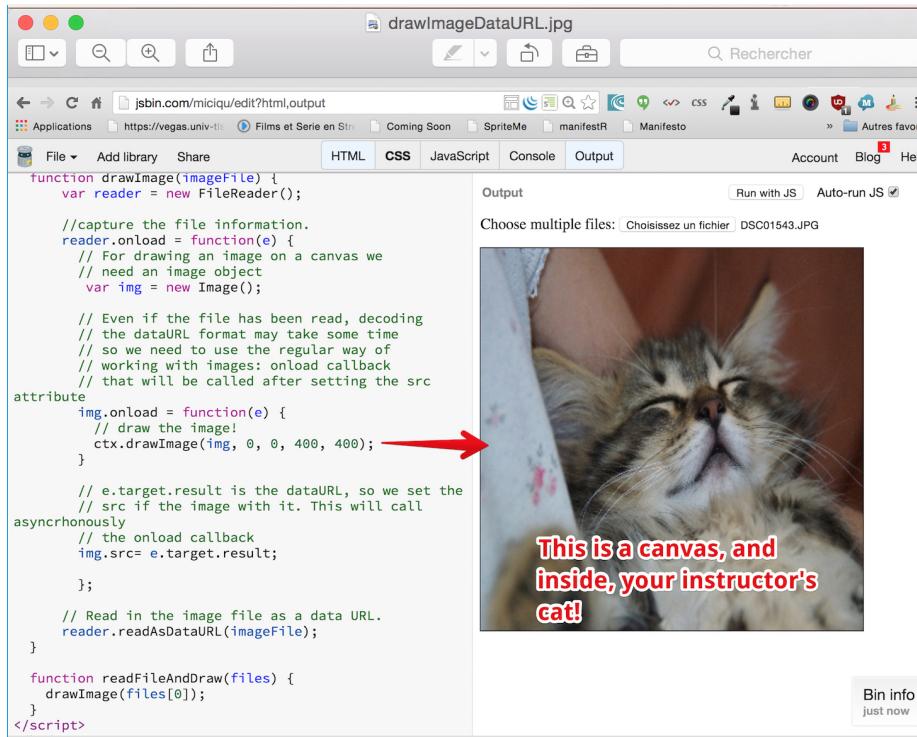
```

Explanations:

- Line 35: starts the reading of the file `f`. When `f` is read, the `onload` callback will be called.
- Lines 25-31: we build, using the DOM API, a `...` and inside we add an `` element with its `src` attribute equal to the url of the image that has been read (the image content as dataURL is in `e.target.result`). Finally, at line 31, we insert the span in the document before the current children of the `<output id="list">` element (declared at line 5).

EXAMPLE 2: READ A SINGLE LOCAL IMAGE FILE AND USE IT WITH DRAWIMAGE IN A CANVAS

[Try it on JS Bin](#)



Errata: the above screenshot says "choose multiple files", but the example only works with a single file.

Source code extract:

```
function drawImage(imageFile) {
    var reader = new FileReader();

    //capture the file information.
    reader.onload = function(e) {
        // For drawing an image on a canvas we
        // need an image object
        var img = new Image();
        10.   // Even if the file has been read, decoding
              // the dataURL format may take some time
              // so we need to use the regular way of
              // working with images: onload callback
              // that will be called after setting the src attribute
        img.onload = function(e) {
            // draw the image!
            ctx.drawImage(img, 0, 0, 400,400);
        }
        20.   // e.target.result is the dataURL, so we set the
              // src if the image with it. This will call
              // asynchronously the onload callback
        img.src= e.target.result;
    };
    // Read in the image file as a data URL.
    reader.readAsDataURL(imageFile);
}
29.   function readFileAndDraw(files) {
    drawImage(files[0]);
}
```

Explanations:

Remember how we worked with images on a canvas. We had to create an empty image object (*line 8*), set the `src` attribute of the image object (*line 23*), then use an `image.onload` callback (*line 15*), and we could only draw from inside the callback (*line 17*). This time, it's exactly the same, except that the URL comes from `e.target.result` in the `reader.onload` callback (*line 23*).

EXAMPLE 3 (ADVANCED): AN INSTAGRAM-LIKE PHOTO FILTER APPLICATION

Another very impressive example, has been developed by @GeorgianaB, a student of the first iteration of this course. This Web application reads local image files, draws them into a canvas element and proposes different filters. This example is given "as is" for those of you who would like to go further. Just click on the link (or on the image below) and look at the source code.

[Try this example online on GitHub](#) (or click the screenshot below)

gianab.github.io/CanvasFilters/

Applications https://vegas.univ-tl... Films et Serie en Str... Coming Soon SpriteMe manifestR Manifesto Autres favoris

▼ Filters

Vintage

▶ Borders

Upload image

Don't forget to download your image

This example by @giorgianaB works on local image files, read as dataURLs and drawn in a <canvas>, then some filters are applied. See the GitHub web site for further explanation

GIF ANIME MICHEL FAIT LA MOUCHE.jpg

680 x 905 Resize

Download