

"Galley Slave"

Acrobat PDF Galleries

Don Lancaster
Synergetics, Box 809, Thatcher, AZ 85552
<http://www.tinaja.com>
don@tinaja.com
(928) 428-4073

Those "galleries" and other HTML photomenu "thumbnail collections" on the web are really popular these days. Giving viewers a quick set of your products or other snapshots they can zoom in on for more detail.

Sadly, most of these suffer from grievous flaws. They are fairly hard to write and even harder to update. They severely limit your control of exactly what your end viewer will see. They are often slow loading. But worst of all, these often involve dozens to hundreds of fragmented file transfers. Giving your ISP fits, clogging up the web, and slowing down your user.

[Adobe Acrobat PDF](#) offers some interesting alternatives to the classic HTML galleries with a number of compelling advantages. A demo sampler of what you can do with Acrobat Galleries appears as our [Bargain Guided Tour](#).

Key points are that [this is a single loading file!](#) Thanks to Acrobat's [byte range retrieval](#), the file can be quickly viewed well before all images fully background load. You get two (or more) clickable areas per thumbnail, one for more printed info, and one to magnify the image for closer viewing. The code is data base driven for easy updates and changes. All the regular text urls look and behave in expected ways. Arrows and scroll bars provide fast nav. And your last page will automatically handle any partial image counts.

Example sourcecode is [found here](#). This "Galley Slave" demo starts off with a [PostScript](#) program using my [Gonzo Utilities](#) you view and alter in your favorite editor or word processor. You then distill your customized file by using [Adobe Acrobat](#) or GhostScript to generate a .PDF file. The .PDF file is then sent to your website at your ISP. The usual free [Acrobat Reader](#) is needed by your end user.

You first create a simple data base of each thumbnail's source file plus its text description, linking url names, image sizes, and any optional id comments. This can be either external or internal to your program. Because PostScript is a general purpose computing language, you could optionally teach a fancier version to read virtually any format data base written in any language.

Here is one possible example of the data textfile format. This file consists of an array of arrays of text data...

```
/galdata [  
  [% data for the first thumbnail  
    (host source filename for the first thumbnail image)  
    (web url for link to the first full image)  
    (web url for link to the first text description)  
    INTEGER value for horizontal number of pixels in the thumbnail  
    INTEGER value for vertical number of pixels in the thumbnail  
  ]  
  .....  
  [% data for the last thumbnail  
    (host source filename for the last thumbnail image)  
    (web url for link to the last full image)  
    (web url for link to the last text description)  
    INTEGER value for horizontal number of pixels in the thumbnail  
    INTEGER value for vertical number of pixels in the thumbnail  
  ]  
] def
```

Or, using some actual data...

```
/galdata [  
  [  
    % location A.0  
    (albradz2.jpg)  
    (sbtesteq.asp)  
    (albrad02.jpg)  
    (Allen-Bradley SLC-100\nPLC Trainer & Programmer)  
    125 113  
  ]  
  .....  
  [% location Z.8  
    (tutenaz2.jpg)  
    (sbphone.asp)  
    (tutena02.jpg)  
    (Internet over\nlive voice phone lines!)  
    116 125  
  ]  
] def
```

Our example sourcecode uses a slightly more complex data format that is page conditional. This requires `/galdata` to defer its execution till run time. Note that the thumb images *must* be locally available during distill time.

The `image` operator is *very* fussy about getting its pixel counts exactly right. If you get a torn or smeared image, check this detail. Thumbnail images are best used 1:1 as larger sizes will ridiculously increase your final PDF file sizes and download times.

External versus internal data can be selected by commenting or uncommenting a line such as...

```
% (C:\\WINDOWS\\Desktop\\tours\\newtour\\gal.dat01.txt) run
```

Suitable code is next generated to provide background art, text and linking and to create the individual gallery boxes in the correct position on the correct page. Dozens of predefined variables are used to control appearance and sizes. These are easily customized.

One of the trickier parts of the code involves converting multiple JPEG files into combined PS images. This can get done by the PostScript `image` operator, preceded by a `DCTDecode` filter. Here is some example code...

```
/imfile imfilename (r) file store      % establish input JPEG read file
/Data {imfile /DCTDecode filter} store % define a data source
<<                                     % start image dictionary
  /ImageType 1                          % always one
  /Width curimagewidth                  % JPEG width in pixels
  /Height curimageheight                % JPEG height in pixels
  /ImageMatrix [ curimagewidth          % scale and position
    0 0 curimageheight neg 0
    curimageheight ]
  /DataSource Data                      % proc to get filtered JPEG
  /BitsPerComponent 8                   % color resolution
  /Decode [0 1 0 1 0 1]                 % per red book 4.10
>>                                     % close image dictionary
image                                   % call the image operator
```

This code is repeatedly called every time a JPEG image is to get inserted into the to-be-distilled `PostScript` code. Note that this *only* works with `.JPG` or `.JPEG` files.

Unlike a HTML gallery, all of the images are *internally* delivered inside a single file. Early images may be viewed immediately while the others invisibly download as a background task.

A second concern involves creating mouseover links in Acrobat files. This gets done following the `ANN` annotation proc of PostScript's `pdfmark` operator. Details are in Adobe Tech Note #5150. And otherwise known as their freebie [pdfmark Reference Manual](#).

Here is some typical url linking code...

```

mark                                % begin a pdfmark data array
/Rect [0 thumbimageheight
thumbboxwidth totalthumbheight]    % llx lly urx ury mouseover
/Border [0 0 0]                     % no color
/Color [0.7 0 0]                    % not used
/Action <</Subtype /URI
      /URI curtexturlname>>         % link url from input data
/Subtype /Link                       % desired pdfmark action
/ANN                                  % annotation type
pdfmark                              % call pdf operator

```

As shown, the program automatically generates page after page of thumbnails. As many as you want for your chosen rows and columns per page. The last page can be partial or full, depending on how the math works out.

There are some tradeoffs. Certain versions of [Netscape](#) will blow up if you try to access a magnified .JPG file from inside of a .PDF file. The workarounds are either to ignore all 17 remaining NetScape users or to put your linked images inside .HTML shell containers.

IE Internet Explorer also has a curious bug: If you link on an inner page of a .PDF file, it often will return you to the first page rather than your chosen one. Use of the .PDF nav arrows and scrolling minimizes this annoyance.

One small detail: You'll normally want your gallery pages to open at 100% percent magnification with a resized window. Otherwise, your thumbnails might distort. To do this, you get into Acrobat. Then select [File --> Document Info --> Open --> 100% magnification](#) and [Resize to Page](#). Then do a [Save As](#).

Additional details on these concepts are found in the [Acrobat](#), [PostScript](#), and [Webmastering](#) library pages of my [Guru's Lair](#) website.

Sourcecode for this document is [Found Here](#).

Additional consulting services available per www.tinaja.com/info01.asp.