Publishing M-Files in MATLAB

Lisa Oberbroeckling

January 9, 2018

Contents

1 Introduction

This document is written to give students and others information on publishing M-files to HTML files using MATLAB's publishing feature. A good reference on publishing M-files is found at MATHWORKS.COM:
`https://www.mathworks.com/help/matlab/matlab_prog/publishing-matlab-code.html`.

This document has many clickable links with it to either specific reference pages or to published webpages for examples. Thus it may be easiest to read this document online.

2 Basic Scripts or M-Files

Here I will explain some of the syntax needed to publish m-files to webpages (HTML files). If you are doing this for an assignment or project, the specific directions of where to save these files, etc. are given separately.

First, you must understand the difference between a BASIC SCRIPT FILE (m-file) and a PUBLISHABLE SCRIPT FILE. A script file is just an ASCII (American Standard Code for Information Interchange, i.e., basic text) file with an extension of `.m`. Within that file are MATLAB commands as if you had typed them in at the command prompt within the Command Window. You can put comments (and should, especially for long files) to better read/debug the file. Comments start with "%". Here is an example (click on the title below to view in the published file in a browser window):

Link 1: basicMFile.m

```
% Example of Basic Script file
% Lisa Oberbroeckling, Spring 2012
clc
x=linspace(-pi,pi);
y=sin(x);
plot(x,y)
% next problem
A=[1 2 3;4 5 6;7 8 9];
B=[A(1,:); -4*A(1,:) + A(2,:); A(3,:)]
B2=[B(1,:); B(2,:); -7*B(1,:) + B(3,:)]
C=[B2(1,:); -1/3*B2(2,:); B2(3,:)]
```

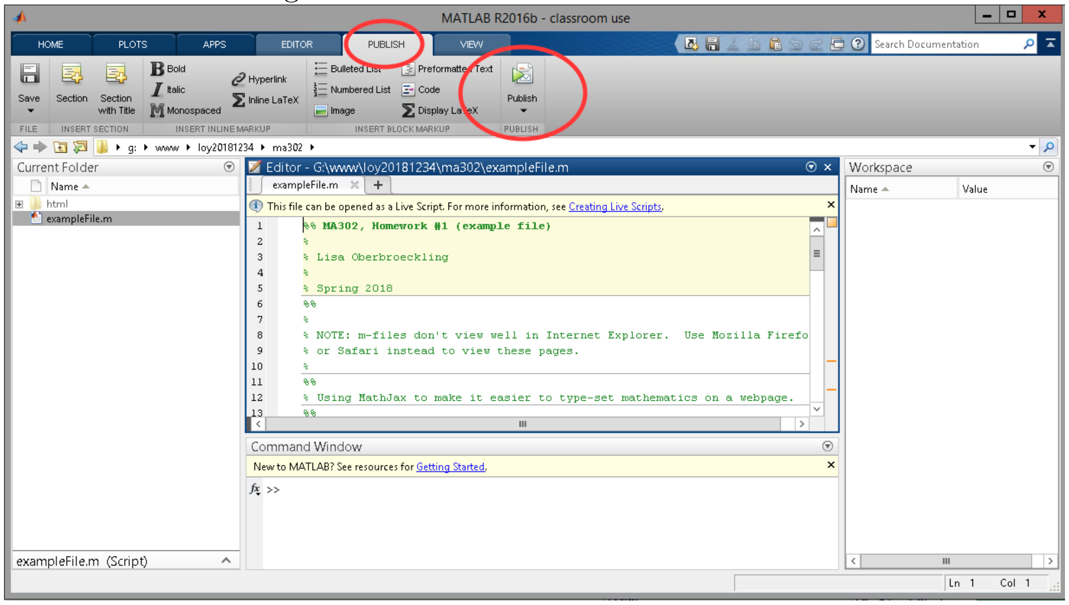
3 Publishing M-files

There are several different ways you can publish your m-files to HTML files using MATLAB. One way is to enter the following on the command line within the Command Window: `publish('filename.m')`.

You can also type: `publish('filename.m', 'html')`. One would do this if you've changed the publishing settings to be "latex", for example.

The other way of publishing, which is more common, is within MATLAB's Editor Window, select the "Publish" tab and press the "Publish" button (see Figure 1).

Figure 1: Publish Tab in Editor Window



This file can be opened as a Live Script. For more information, see Creating Live Scripts.

```
%% MA302, Homework #1 (example file)
%
% Lisa Oberbroeckling
%
% Spring 2018
%%
%
% NOTE: m-files don't view well in Internet Explorer.  Use Mozilla Firefo
% or Safari instead to view these pages.
%
%%
% Using MathJax to make it easier to type-set mathematics on a webpage.
%%
```

New to MATLAB? See resources for Getting Started.

```
>>
```

This will create a folder named "html" in the current working folder (if there isn't a folder of that name already there), and put the **`filename.html`** file and any other files necessary for the webpage (PNG files for images, for instance) in the "html" folder. When we publish the above file, it won't be very pretty. See Link 1:
`http://math.loyola.edu/~loberbro/matlab/v2018/html/basicMFile.html`.

Notice that the output that is shown on the webpage is out of order of the commands given; the output of the next problem appears BEFORE the plot. Thus we want to format the comments in a special way so that when it is published, the MATLAB commands and output appear in order. This is the topic of the next section.

4 Using Sections (Formerly Cells)

In order to format our m-file to make it better for publishing, we want to break up the commands in the m-file into SECTIONS. Each SECTION is broken up by "`%%`". When you do this within the Editor Window, you'll notice lines appearing between each section. Sections are useful not only for publishing, but for running and debugging scripts. Sections are used in publishing to signify different sections of the webpage (like for homework assignments to have different sections for each problem). Sections also determine how/where output for

lines of code are displayed. For more detailed information, go to
`http://www.mathworks.com/help/matlab/matlab_prog/run-sections-of-programs.html`.

4.1 Using sections for publishing

If you have text following the “`%%`” on the same line, this also creates a SECTION with that text as the section title. In addition, a bulleted list is created with those linked section titles at the top of the webpage. For the following we just added “`%%`” to two lines (lines 1 and 7); compare with basicMFile.m on page 2.

Link 2: publishMFile1.m

```
1  %% Example of Basic Script file
2  % Lisa Oberbroeckling, Spring 2012
3  clc
4  x=linspace(-pi,pi);
5  y=sin(x);
6  plot(x,y)
7  %% second problem
8  A=[1 2 3;4 5 6;7 8 9];
9  B=[A(1,:); -4*A(1,:) + A(2,:); A(3,:)]
10 B2=[B(1,:); B(2,:); -7*B(1,:) + B(3,:)]
11 C=[B2(1,:); -1/3*B2(2,:); B2(3,:)]
```

The above is better than the published page without sections, but can be better. We may want the page to start with a title. This is done by adding a line “`%%`” after our title (and another comment line(s) for other introductory text, like my name).

LINK 3: publishMFile2.m

```
1  %% Example of Basic Script file
2  % Lisa Oberbroeckling, Spring 2012
3  %%
4  clc
5  x=linspace(-pi,pi);
6  y=sin(x);
7  plot(x,y)
8  %% second problem
9  A=[1 2 3;4 5 6;7 8 9];
10 B=[A(1,:); -4*A(1,:) + A(2,:); A(3,:)]
11 B2=[B(1,:); B(2,:); -7*B(1,:) + B(3,:)]
12 C=[B2(1,:); -1/3*B2(2,:); B2(3,:)]
```

If you look at the published webpage, you’ll notice that we have a section link and title for the “second problem” but not for the first. So we probably want to change line 3 to include a section title:

LINK 4: publishMFile3.m

```
%% Example of Basic Script file
% Lisa Oberbroeckling, Spring 2012
%% first problem
clc
x=linspace(-pi,pi);
y=sin(x);
plot(x,y)
%% second problem
A=[1 2 3;4 5 6;7 8 9];
B=[A(1,:); -4*A(1,:) + A(2,:); A(3,:)]
B2=[B(1,:); B(2,:); -7*B(1,:) + B(3,:)]
C=[B2(1,:); -1/3*B2(2,:); B2(3,:)]
```

When publishing m-files, each time a new section is started, MATLAB displays the output created by the commands of the previous section. The difference between `publishMFile3.html` and `publishMFile4.html` is where the output is displayed for the second problem.

LINK 5: publishMFile4.m (partial view)

```
%% second problem
% problem 2a
A=[1 2 3;4 5 6;7 8 9];
B=[A(1,:); -4*A(1,:) + A(2,:); A(3,:)]
%%%
% problem 2b
B2=[B(1,:); B(2,:); -7*B(1,:) + B(3,:)]
%%
% problem 2c
C=[B2(1,:); -1/3*B2(2,:); B2(3,:)]
%%
% problem 2d
x=linspace(-10,10);
y=exp(x);                 % can comment after a command, too
plot(x,y)
title('Another Example')
```

Another important place to insert a section break is when you want to have text following Matlab commands, but within the same section. If you just include comments after the Matlab commands, they will be formatted as comments within the displayed code, not as text. Instead, insert a section break (without a section title) and then the comment block that will be the text:

LINK 5: publishMFile4.m (partial view)

```
%% third problem
A=[1 2 3;4 5 6;7 8 9;eye(3)]
% If I don't have a cell break above this comment, this
% text just appears as comments within the command lines,
% and not text on the webpage.
%%
% Instead, have a cell break or a section break
```

Note that you can also have section titles without having section breaks. This is done by having the line start with `%%%`" along with the section text. This will have the text and/or Matlab commands be within the sections, but the output of those commands at the next section break, which may not have the desired effect.

LINK 5: publishMFile4.m (partial view)

```
%%% Next section
% this section does not have a cell break.  This may
% or may not be useful depending on how you want the
% output displayed on the published webpage. It works
% here because this is the last section and cell.
x=linspace(0.0001,10);
y=log(x);
plot(x,y)
```

Using sections is especially important for m-files with multiple plots. Remember, MATLAB only shows the last plotting command (like `plot`, ... `mesh`, `surf`, etc.). You can have multiple commands appear on the same figure by using the `hold on` and `hold off` commands. But if you want to display multiple figures (not in the same window), you have to either use the `pause` command of the `figure` command. This first example uses the `pause` command:

LINK 6: publishMFile5.m

```
%% Example of Basic Script with pause
% Lisa Oberbroeckling, Spring 2012
%%
x=linspace(-pi,pi);
y=sin(x);
plot(x,y)
hold on
y=cos(x);
plot(x,y,'r')
hold off
title('First Plot')
pause
[x,y]=meshgrid(linspace(-10,10));
z=sin(x).*cos(y);
mesh(x,y,z)
xlabel('x'),ylabel('y'),zlabel('z')
title('Second Plot')
```

If you run the file publishMFile5.m, the first figure will appear and then Matlab will be paused. The second will appear after any key is pressed. When this is published, even the publishing will be on pause after the first figure is created until you press a key. But if you look at the webpage, only the last figure is actually shown on the webpage. As discussed above, when publishing, Matlab runs each section as a block and then displays any output. At the end of the section the only output that Matlab sees as being created by the section of commands is the second figure. The second figure replaces the first figure, so it is not shown on the webpage. Thus, we need to create a section for each figure we want on the webpage. When we publish the m-file, we have to remember to "press any key" before the publishing can continue, which is really annoying (especially when you forget/don't know there is a `pause` there!) so you may want to comment the `pause` command out or take it out completely. In the following example we created a new section without a section title.

LINK 7: publishMFile5b.m

```
%% Example of Basic Script with pause
% Lisa Oberbroeckling, Spring 2012
%%
close all
clc
% first plot
x=linspace(-pi,pi);
y=sin(x);
plot(x,y)
hold on
y=cos(x);
plot(x,y,'r')
hold off
title('First Plot')
% pause
%%
[x,y]=meshgrid(linspace(-10,10));
z=sin(x).*cos(y);
mesh(x,y,z)
xlabel('x'),ylabel('y'),zlabel('z')
title('Second Plot')
```

The next group of files use the `figure` command.

LINK 8: publishMFile6.m

```
%% Example of Basic Script with figure command
% Lisa Oberbroeckling, Spring 2012
%%
close all
clc
% first plot
figure(1)
x=linspace(-pi,pi);
y=sin(x);
hold on
```

```
y=cos(x);
plot(x,y,'r')
hold off
plot(x,y, 'r')
title('First Plot')
% Second plot
figure(2)
[x,y]=meshgrid(linspace(-10,10));
z=sin(x).*cos(y);
mesh(x,y,z)
xlabel('x'),ylabel('y'),zlabel('z')
title('Second Plot')
% third plot
figure(3)
[x,y]=meshgrid(linspace(-10,10));
z=x.*cos(y);
mesh(x,y,z)
xlabel('x'),ylabel('y'),zlabel('z')
title('Third Plot')
% fourth plot
figure(4)
[x,y]=meshgrid(linspace(-10,10));
z=x.*y;
mesh(x,y,z)
xlabel('x'),ylabel('y'),zlabel('z')
title('Fourth Plot')
```

When the above file is published, it puts each figure side-by-side on one line. Depending on how many figures you have this may not have the desired effect, so you may want to have each figure within its own section instead, like in `publishMFile6b.html` below.

LINK 9: publishMFile6b.m

```
%% Example of Basic Script with figure command
% Lisa Oberbroeckling, Spring 2012
%%
close all
clc
%% first plot
figure(1)
x=linspace(-pi,pi);
y=sin(x);
hold on
y=cos(x);
plot(x,y,'r')
hold off
plot(x,y, 'r')
title('First Plot')
%% Second plot
figure(2)
[x,y]=meshgrid(linspace(-10,10));
```

```
z=sin(x).*cos(y);
mesh(x,y,z)
xlabel('x'),ylabel('y'),zlabel('z')
title('Second Plot')
%% third plot
figure(3)
[x,y]=meshgrid(linspace(-10,10));
z=x.*cos(y);
mesh(x,y,z)
xlabel('x'),ylabel('y'),zlabel('z')
title('Third Plot')
%% fourth plot
figure(4)
[x,y]=meshgrid(linspace(-10,10));
z=x.*y;
mesh(x,y,z)
xlabel('x'),ylabel('y'),zlabel('z')
title('Fourth Plot')
```

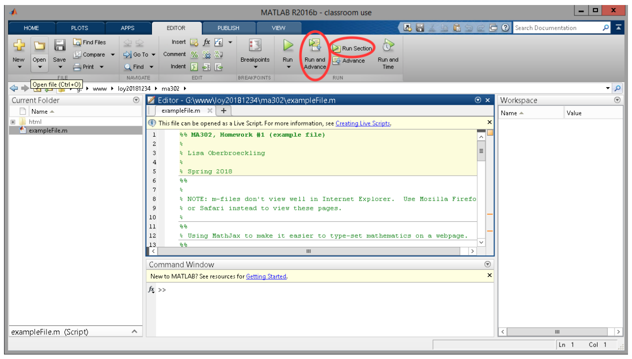
The files `publishMFile6c.html` and `publishMFile6d.html` provide other alternatives to formatting the section breaks depending on how you want to organize the code and graphics.

4.2 Using sections for running/debugging files

Note that you can also use sections for running/debugging code and can be very useful. This topic is only briefly covered in this document. You can separate out self-contained portions of your code and just run that piece. This is good for long assignments within one file; just run on problem at a time to see if runs as expected. You can run the code within a section block several ways.

1. While the cursor is within the section block you want to run, use the keystrokes of Ctrl-Enter (Cmd-return on a Mac). Using the keystrokes of Ctrl-Shift-Enter (Mac: Cmd-return-enter) will run the current section and advance the cursor within the next section block.

2. Select "Run Section" (or "Run and Advance") in the "Editor" Tab (shown below).



This file can be opened as a Live Script. For more information, see Creating Live Scripts.

```
%% MA302, Homework #1 (example file)
%
% Lisa Oberbroeckling
%
% Spring 2018
%%
%
% NOTE: m-files don't view well in Internet Explorer.  Use Mozilla Firefo
% or Safari instead to view these pages.
%
%%
% Using MathJax to make it easier to type-set mathematics on a webpage.
%%
```

New to MATLAB? See resources for Getting Started.

WARNING: when you run (evaluate) a section block, the file is not saved automatically as it is when you run the entire m-file! Also, any variables used within that section must already be assigned.

5 Formatting Text

You can format your m-file by clicking on various options on the “Publish” tab within MATLAB (must have a script file open before seeing this tab). By clicking on one of the items on the tab, Matlab will insert text into your m-file for that purpose. This includes inserting text for things already discussed above, like inserting a section break, title, etc. You also have the ability to customize your menu/toolbar to add others.

Using the buttons on the toolbar can take extra time after awhile, so it is also useful to know how to just type in the formatting. Note that these are presented not in the order of the toolbar.

5.1 Basic text formatting

In order to create a new line or new paragraph of text, have a blank comment line in between the lines.

Link 10: publishMFile7.m (partial view)

```
%% Basic text formatting
%
% In order to create a new line or new paragraph of text,
% have a blank comment line in
% between the lines,
%
% Here is a paragraph that just goes on,  and on, and on,
% and on, and on, and on, and on, and on, and on, and on,
% and on, and on. Still going on,
% and on, and on, and on, and on, and on.
%
% Then we have a new line/paragraph.
%
```

Text can be formatted to be **bold**, *italic*, `monospaced`, or combinations such as ***bold and italic***.

Link 10: publishMFile7.m (partial view)

```
% Text can be *bold*, _italic,_ and/or |monospaced|.
% One can also combine these formats like:
```

```
%
% _*BOLD, ITALIC TEXT*_
%
% |_ITALIC, MONOSPACED TEXT_|
```

5.2 Lists

One can have an unordered, or bulleted list.

Link 10: publishMFile7.m (partial view)

```
%% Unordered (Bulleted) List
%
% * first item
% * second item blah blah
%
```

Keep in mind that you must have a section break before the list, with or without a section title. You also must have a blank comment line to end the list. Here's another example.

Link 10: publishMFile7.m (partial view)

```
%%
%
% * item number 1
% * item number 2
%
```

You can also have an ordered (numbered list) using the same formatting as above, but with "#" instead of "*" for each list item.

Link 10: publishMFile7.m (partial view)

```
%% Ordered (Numbered) List
%
% # first item
% # second item blah blah
%
```

As in the bulleted list, one must have a section break, with or without a section title. Second example without section title:

Link 10: publishMFile7.m (partial view)

```
%%
```

```
%
% # blah blah blah blah blah blah blah blah blah blah blah
    blah blah blah blah blah blah blah blah blah
% # yadda yadda yadda yadda yadda yadda yadda yadda yadda
    yadda yadda yadda yadda yadda yadda yadda yadda yadda
    yadda yadda
%
```

5.3 HTML Links

You can have the links display the URL or display other text. URL as the link:

Link 10: publishMFile7.m (partial view)

```
% Here's an example of using the URL as
% the text of the link: <http://www.mathworks.com>
```

You can have any text for the link:

Link 10: publishMFile7.m (partial view)

```
% Here's an example of having other text as the link:
% <http://www.mathworks.com MathWorks>
```

LINKING THE M-FILE: It may be nice (required!) to link the m- file that was published to create the webpage. The easiest way to do it is like it as a relative path rather than using the absolute path (URL).

Link 10: publishMFile7.m (partial view)

```
% The easiest way to do it is like this
% <../publishMFile7.m M-File for this page>.
```

The "`../`" before the filename means to go back one folder from where the HTML file is located, which is where the m-file is located.

IMPORTANT: the text within the "<" and ">" must be on the same line within the file. The editor might automatically wrap the text if the URL and/or the text for the link is long. If this is the case, you must go back and make it one line. Otherwise, the link will be broken!

5.4 Inserting images

Any figures that the MATLAB code creates will automatically be saved as PNG files and inserted on the webpage. You can also include images like the following example.

Link 10: publishMFile7.m (partial view)

```
%
% <<peaks.jpg>>
%
```

The image must be on its own line; no text can appear before or after it for the image to be shown correctly on the page. Also note that the above is assuming the file for the image is located in the "html" folder into which the HTML file the M-file produced is located.

5.5 Preformatted text and code

Preformatted text is a way to display text on the webpage EXACTLY as it appears in the editor, including extra spaces, line breaks in exactly the same place, etc. This is commonly used for displaying lines of code in programming, although the "Code" formatting is better.

IMPORTANT: notice that the code below doesn't appear any different than plain comments within the m-file. In order for it to be preformatted, you must use the menu item to insert the lines and change it. Here is what the menu item inserts:

Link 10: publishMFile7.m (partial view)

```
%
%  PREFORMATTED
%  TEXT
```

Then you change the words "PREFORMATTED" AND "TEXT" and add lines if necessary:

Link 10: publishMFile7.m (partial view)

```
%  preformatted text
%   displayed
%  exactly
%  as it appears   in      the    editor
%  (commonly used to display lines of code but is better to
   use Code)
%  ------------------------------------------------------
%  function y = myexample(x)
%  % MYEXAMPLE(X) is a function for example purposed only
%  %
%  y = x.^2;
% ------------------------------------------------------
```

Similarly, when you select "Code" it adds the lines

Link 10: publishMFile7.m (partial view)

```
%
%   for x = 1:10
%       disp(x)
%   end
```

and you change the selected code to your own:

Link 10: publishMFile7.m (partial view)

```
%
%   function y = myexample(x)
%  % MYEXAMPLE(X) is a function for example purposed only
%  %
%  y = x.^2;
```

Notice how the coloring is better using Code rather than Preformatted Text! This "Code" setting should be used whenever you want keywords, comments, etc. in the code to be colored appropriately rather than using "Preformatted Text".

5.6 Inserting HTML code

You can insert other HTML code (such as tables) into your document and it will treat it as regular HTML code when published as an HTML file.

Link 10: publishMFile7.m (partial view)

```
% <html>
% <table border="1" cellspacing="0" cellpadding="3"><tr><td>
    one</td><td>two</td></tr></table>
% </html>
```

5.7 Inserting LaTeX equations

Basic LaTeX equations can be displayed on the webpage if the mathematics are contained within "`$$`". Only basic mathematics can be displayed when publishing to HTML mode, and some symbols don't display correctly. For example, the not-equals symbol "$\neq$" didn't work in earlier versions, then it worked in version 2012a, then didn't work in version 2013a, and now appears to work in 2016a. Technically speaking, when these equations are published to an HTML file, the equation is saved as a PNG image (in the “html” folder or same folder the HTML file is saved) and that image file is displayed on the webpage. This makes the webpage slower to load, and not as accessible for screenreaders, etc. Use MathJax instead.

Link 10: publishMFile7.m (partial view)

```
% One can insert basic LaTeX commands and equations between 2
    dollar signs.
% One can have LaTeX code inline like $$ e^{\pi i} + 1 = 0 $$
% or on a separate line.
%
% $$ y_1 = \frac{d}{dx}\int_0^{\pi x} e^{\theta^2} d\theta $$
```

These images make the formulas appear blurry and/or small, and as mentioned above, not every symbol will display correctly. Thus it may be better to use MathJax instead.

In order to use MathJax, you must insert the following line(s) into the m-file:

Code to use MathJax

```
% <html>
% <script src='https://cdnjs.cloudflare.com/ajax/libs/mathjax
   /2.7.2/MathJax.js?config=TeX-MML-AM_CHTML'></script>
% </html>
```

When you use MathJax, you put inline equations within “\(” and “\)” and for displayed equations you put them within “\[” and “\]” as shown below.

Link 10: publishMFile7.m (partial view)

```
% Here are the same mathematics as in the previous section
   now displayed with MathJax: \( e^{\pi i} + 1 = 0 \)
%
% \[ y_1 = \frac{d}{dx}\int_0^{\pi x} e^{\theta^2} d\theta \]
```

Notice how much better the mathematics typeset using MathJax are, and they are NOT images so are much more accessible!

5.8 M-file and HTML file for this section

The m-file that includes examples of all of this text formatting:

`http://math.loyola.edu/~loberbro/matlab/v2018/publishMFile7.m`

HTML published file:

http://math.loyola.edu/~loberbro/matlab/v2018/html/publishMFile7.html

6 Summary of M-Files/Links

All of the following m-files can be found at:
http://math.loyola.edu/~loberbro/matlab/v2018.

The published webpages can be found at:
http://math.loyola.edu/~loberbro/matlab/v2018/html/.

1. `basicMFile.html`: example of a basic M-file or SCRIPT file.
2. `publishMFile1.html`: example of minimally changing basicMFile.m for publishing.
3. `publishMFile2.html`: example of putting in a title and introductory text.
4. `publishMFile3.html`: adding section title.
5. `publishMFile4.html`: examples of placing section breaks for displaying output.
6. `publishMFile5.html`: basic example of plots and the `pause` command.
7. `publishMFile5b.html`: modifiying output of plots using section breaks.
8. `publishMFile6.html`: basic example of plots and the figure command.
9. `publishMFile6b.html`: modifying output of plots using section breaks and figure command.
10. `publishMFile7.html`: examples of formatting text.