```
#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
# Various ESSENTIAL Mini-How-To's
#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Topics:
```

```
#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
A.  Overview
#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
# Generally, to modify DLMF's source files, you'll need
#   * a checkout of DLMF's sources
# To build a pdf of the Handbook
#   * a recent texlive installation
# To build the website
#   * LaTeXML
#   * a working build directory
# To Test the website
#   * a tomcat installation
#   * optionally apache httpd
# See the following sections for more details.

#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
B. Cleanup
#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
# Note that the dlmf materials are BIG!
#   * dlmf cvs checkout: ~4.5Gig   in ~/dlmf
#   * build directory  : ~4.5Gig   in orion:/local/dlmf/$USER/dlmf
#   * war file         : ~1.0Gig   in orion:/local/dlmf/$USER/*.war
#
```

```
# So, if you are regularly working with dlmf it makes sense to keep
# the files available.  But if you will not be modifying or building
# for a "long" time, especially if it is just a one-shot experiment,
# you really should remove materials you won't need.

#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C. Working with DLMF's repository
#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
# Currently all of DLMF's files are stored in a CVS repository on
#  math-idev.cam.nist.gov.

#========================================================================
#     1. Checking out DLMF from its cvs repository
#========================================================================
               This step is to get a copy of the DLMF files
               in your directory--Its a checkout using CVS

# Before you do anything, you'll need all the DLMF documents, programs
# and data.  I'll assume that you run this in your home directory
# so that all files end up under ~/dlmf

cd ~
cvs -d :ext:math-idev.cam.nist.gov:/local/cvs/dlmf checkout dlmf

# Note that this currently contains ~4.5GB of data
# so consider if you need it, and where you put it!

#========================================================================
#     2. Updating DLMF
#========================================================================
# Before carrying out any of the following operations,
# such as building the site, or committing your own changes,
# consider that you may want to update your local copy of dlmf
# to pick up any recent changes that others have made.

cd ~/dlmf
cvs -q update

# This will update any files that were committed since the time
# you last updated (shown prefixed with "U" or "P"),
# or will merge those changes with any you have made. If there
# is a conflict with those 2 sets of changes, it will prefix with
# "C" and you will need to find sections in those files marked like
#    <<<<<<  (filename)
#    your changes
#    ======
#    other changes
#    >>>>>> (latest revision number)
# You'll have to edit and figure out which changes are best
# before committing your changes.
#
# Otherwise, files that are locally modified will be marked with "M";
# those are files you'll want to commit (see below).

#=======================================================================
#     3. Comparing changed files
#=======================================================================
# Now that you've modified a file (or fixed conflicts), you'll want
# to commit it to the repository.  But first, you'll probably want
# to verify that you've changed what you wanted:

cd ~/dlmf
cvs diff whatever/got/changed

# will show those changes in a diff like format.

#=======================================================================
#  4. Committing changed files
#=======================================================================
# Now, that you've verified the changed files, and ideally after verifying
# that they are valid, that the site and/or book still compile,
# you will want to commit those changes to the repository:

cd ~/dlmf
cvs -q update
cvs commit -m "comment about what changed and why" whatever/got/changed

#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
# D. Getting & Updating LaTeXML from it's repository.
#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
# LaTeXML is currently stored on GitHub (https://github.com).
# You can find out about installing & running LaTeXML in general at
#     http://dlmf.nist.gov/LaTeXML/
#
# Note that when we build on orion, we're NOT using the latexml in
# my home directory, but a separate "safe" checkout on orion at:
#   /local/dlmf/LaTeXML.
# I usually will have updated it when I'm convinced that it is "safe";
# so these sections are for an overview.

#=======================================================================
#     1. Checking out LaTeXML
#=======================================================================
# To check out a copy of LaTeXML from github, do:
                   How to get your own copy of LaTeXML
cd /local/dlmf
git clone https://github.com/brucemiller/LaTeXML.git
cd LaTeXML
perl Makefile.PL
make
make test

#=======================================================================
#     2. Updating LaTeXML
```

```
#=========================================================================
# When building the website, you may want to update LaTeXML first
# (if you have write permissions)

cd /local/dlmf/LaTeXML
git pull
# if there were _added_ files, run this:
perl Makefile.PL
# If there were any changes, run these:
make
make test

#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
# E. To make Book.pdf
#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
# Note that you may want to update your dlmf checkout; see above;
# Note that you don't need LaTeXML.
# You will need to be on a machine with a relatively recent TeX-live;
# eg Fedora, NOT Centos (or Scientific-Linux?)
#  --- try hyacinth.cam.nist.gov

#=========================================================================
#     1.Basics
#=========================================================================
# Assuming that ~/dlmf/bin is in your path, you simply need to run:

cd ~/dlmf
DLMFtex book

# If things really get screwed up, try removing all *.aux files, then retry.
# Also, if it appears to hang, it may be that DLMFtex is inadvertently
# hiding an error message (it tries to hide the voluminous output of latex)
# use the "-v" option to let it print all that stuff out.

# IMPORTANT!!!
# If you have been modifying the chapter content and see a message
# from DLMFtex like:
#   Normally the label=>refnum associations should NOT change
#   ...
# ESPECIALLY watch for
#    "The following labels are now missing"
# and
#    "The following labels have changed refnums"
# PLEASE consult the following subsection.

#=========================================================================
#     2.Updating the Labels
#=========================================================================
# It is IMPORTANT that every chapter, section, equation, table,...
# is PERMANENTLY associated with a specific reference number
# so that people can safely refer to Equation 1.2.3.
#
```

```
# This means that additions have to be made only in places
# that don't affect the numbering of following material
# (such as at the end of sections).  And deletions either
# have to only delete non-numbered material, or leave
# "stubs" that preserve the number of the missing object.

# To monitory this, we keep a record of the internal labels
# (used in latex \label) and thier associated reference numbers
# (the visible number) in the file
#      ~/dlmf/etc/labels.fixed
# which is kept under CVS.  When running DLMFtex, the new
# set of associations are compared to the fixed one and
# reports the differences.
#
# Thus, if you have been modifying the chapter content and see
# a message from DLMFtex like:
#   Normally the label=>refnum associations should NOT change
#   ...
# ESPECIALLY watch for
#    "The following labels are now missing"
# and
#    "The following labels have changed refnums"
# You should VERY CAREFULLY examine the modified sources, the
# message and output to determine if it is caused by inappropriately
# placed insertions or deletions.  Insertions _can_ be made
# in such a way to not cause renumbering of following material.
#
# If the changes ARE appropriate (typically should only be
# additions), then you will want to update the record and commit
# it to CVS:

cd ~/dlmf
cp labels.tmp etc/labels.fixed
cvs commit -m "Explanation here" etc/labels.fixed

# BUT PLEASE: work with me or someone else knowledgeable on this!!!!

#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

## F. **Build a Draft DLMF Website**

```
#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
# We'll first go through building a draft version of the website.
# Building a formal release (see below) uses many of the same steps,
# and generally you will want to have tested a draft first, anyway.
#
# First consider whether you need to update DLMF or LaTeXML (see above):
#
# In the common case, you will ssh onto orion to do the builds.
# If you haven't already done this, you should copy my local.conf
# file into your ~/dlmf/etc

   cp ~miller/dlmf/etc/local.conf ~/dlmf/etc/local.conf
```

```
# ALSO, we will now be packaging up a draft version of DLMF
# to be placed on the external server, but visible under
#  http://dlmf.nist.gov/draft/

#======================================================================
#     1.Basic building
#======================================================================
# The basic command for building the DLMF website is:

makesite [options]

# This will convert the TeX and other sources in ~/dlmf
# into a tomcat webapp in a directory on orion at
#    /local/dlmf/$USER/dlmf
# where $USER is your username.
#
# makesite attempts to work like "make" in that it will build the
# entire site from scratch if it has not yet been built
# (which may take 1-2 hours), but generally will only reprocess
# files that have changed since the last time you ran it,
# which goes much quicker.
#
# Sometimes it overlooks indirect dependences, however,
# and some of the options described below will force it to
# As a last resort, removing all files under /local/dlmf/$USER/dlmf
# will make it build from scratch, which always should work.
#
# Note also that in order for makesite to create errata.pdf,
# you should have made a book.pdf at least once
# in order for the appropriate *.aux files to be present.

#======================================================================
# Important makesite options:
#======================================================================

# You can get a (brief) overview of all options by running:

makesite --help

# When you know you've got new or significantly changed citations
# (so it knows it needs to rebuild the bibliography)

   --force=scan --force=paginate

# that causes it to clear out and rebuild the database
# (which records all labels, references, citations, cross links, etc)
# Since it causes the xml pages to be remade, you'll automatically get
# the html pages remade.  Otherwise, continue on.

# To force the (various kinds of) html pages to get remade
# (but not necessarily preceding computations)
```

```
   --force=instanciate

#=====================================================================
# Publishing the draft: How to make the .WAR file and push it out
#=====================================================================
# After you've built dlmf and verified it looks good, you
# may want to publish it to
#   http://dlmf.nist.gov/draft/
# so that Adri can see it, or that we all can have access from home
# to test using different OS or browsers.

# First, create a war (Web ARchive) file by running

makesite war

# This takes a few minutes and will create the file:
#   /local/dlmf/$USER/dlmf-draft.war
# You can publish this to dlmf.nist.gov (aka muggle) bu running:

push-dlmf-draft /local/dlmf/$USER/dlmf-draft.war

# (where you, of course, substitute your username for $USER)
# After it has finished copying, and after a minute, the draft
# should be visible at
#    http://dlmf.nist.gov/draft/
#
# Note that when a draft is near ready for release, you probably
# should go ahead and prepare news items, errata and set the
# RELEASE_VERSION, and RELEASE_DATE as described in the next section.

#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
# G. Make a new public Version or Release:
#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
# We will assume that you have built and tested a Draft version of
# the site before proceeding to make a public release.

#=====================================================================
# 1. Preliminaries
#=====================================================================
# Before building you will want to take several steps:
#  (1) Choose a VERSION_NUMBER and RELEASE_DATE;
#      Think of the version number as:
#          Edition.Printing.Update
#      except that Printing and Update start from 0.
#      For typical updates, we just increment the 3rd number.
#      Likewise, choose a release date, typically a couple of
#      days in the future, to allow for final testing, announcing, whatever.
#
#      Before building, update
#          dlmf/etc/dev.conf
#      (or whichever conf you're using)
#      to modify the values of "version " and "timestamp"
#      Also make sure the date & release used in the errata
#      and news item match!
#
#  (2) If it is worth making a release, it is worth explaining why;
#      Add a short note about the release by prepending an item to
#          dlmf/about/news/index.tex
#      That item should mention the release version and date.
#      If the changes are non-trivial, you should include a link to
#      the errata like
#          see \longref{errata:VERSION_NUMBER} for details
#      substituting the actual version number for VERSION_NUMBER.
#
#  (3) A new release should have a new section in the errata,
#      even if only to say "Several minor improvements were made.".
#      Prepend a section to
#          dlmf/front/errata.tex
#      See the file for format and examples; in particular, be sure
#      to use the VERSION_NUMBER and RELEASE_DATE in the section title,
#      and add a label:
#        \label{errata:VERSION_NUMBER}
#  (4) Refresh book.pdf (see above), if necessary so the *.aux files are fresh;
#      these are used in making errata.pdf!
#
# Also, you obviously want to update your dlmf from CVS,
# but you should also be sure to commit all local changes, as well.

#======================================================================
# 2. Building the Public Release
#======================================================================
# In order to be sure that the release is "clean & fresh",
# I typically remove the previously made draft before building:

rm -rf /local/dlmf/$USER/dlmf

# By default, you'll get various "DRAFT" indicators in the webpages
# the --nodraft option eliminates this.

makesite --nodraft war

#[ASIDE: you _could_ avoid removing & rebuilding from scratch by
# replacing the previous 2 command with the single:

makesite --force=instanciate --nodraft war
# ]

# This should create a fresh version of the site on orion in
#   /local/dlmf/$USER/dlmf
# as well as a new war file at
#   /local/dlmf/$USER/dlmf-YYYYMMDD.war
# where YYYYMMDD is be the release date.
# You can make a final test it out on
      http://orion.cam.nist.gov/dlmf/
# (or wherever your builds are usually seen)

#=======================================================================
# 3. Publishing the release
#=======================================================================

#=======================================================================
#            Publish on dlmf.nist.gov
#=======================================================================
####################################################
# For security, these steps can only be done by:  #
#    Chris Schanzle <schanzle@nist.gov>           #
#    Don Koss <donald.koss@nist.gov>              #
####################################################
# But I'll outline them for the overview.
# Carry out the same steps as for testing on dlmf-dev,
# except use host muggle.nist.gov (the actual host name of dlmf.nist.gov)
#
# Ideally this should be done on BOTH math-dev.nist.gov
# and muggle.nist.gov (which serves dlmf.nist.gov)

# Copy the war file to the server.

scp dlmf-YYYYMMMDD.war miller@muggle.nist.gov:/local/home/miller/

# Login & install it

ssh miller@muggle.nist.gov
sudo chown mcsdweb:mcsdweb dlmf-YYYYMMDD.war
sudo cp /local/home/miller/dlmf-YYYYMMDD.war /local/tomcat5

# and get it running

sudo rm /local/tomcat5/webapps/dlmf.war
sudo ln -s /local/tomcat5/dlmf-YYYYMMDD.war  /local/tomcat5/webapps/dlmf.war

# and probably should remove the local copy, so it doesn't waste backups...
rm /local/home/miller/dlmf-YYYYMMDD.war

# After a minute or so, tomcat should have rescanned the war file, so
#  TEST IT!!!
# http://dlmf-dev.nist.gov/ or
# http://dlmf.nist.gov/

#=======================================================================
# 4. Bookkeeping for Posterity
#=======================================================================
# After you're sure everything actually works,
# make sure you've saved everything back in cvs (see above),
# you'll want to do some bookkeeping so that later on we
# can figure out what a particular release was made of.

# Store a copy of the war file on orion in /local/dlmf/archive

cp /local/dlmf/$USER/dlmf-YYYYMMDD.war /local/dlmf/archive

# AND, you should "tag" the current revisions of everything
# in dlmf as being associated with the current VERSION_NUMBER!
# Given that the VERISON_NUMBER is of the form EDITION.PRINTING.UPDATE,
# use the following command to tag the current set of files:

cvs tag -R dlmf-EDITION-PRINTING-UPDATE

# (ie. substitute "-" for ".")

# Or if a couple of days have past, use the remote, dated form:
cvs rtag -D YYYY-MM-DD dlmf-EDITION-PRINTING-UPDATE dlmf

# If you were making another printing or edition of the printed book,
# you should also mark the version using:

cvs tag -R hmf-EDITION-PRINTING-UPDATE

# where hmf stands for "Handbook of Mathematical Functions"

#----------------------------------------------------------------------
# Incidentally:
#   You can list the tags on a specific file by doing, eg.:
cvs status -v book.tex
#   You can check out a copy of everything as it was
# for a particular tagged version, say, 1.0.2, by doing:
cvs checkout -r dlmf-1-0-2

# Or, to compare the current revision of a file with the
# way it was in version 1.0.2, you can say
cvs diff -r dlmf-1-0-2 somefile

#----------------------------------------------------------------------
# As an aside, LaTeXML is in an git repository, not cvs,
# and so the tagging setup is different.
# Actually, I need to check how to do tagging in git.

# For svn, it was the following:
# Go to the svn/LaTeXML directory (where you'll find ./trunk)
# copy the current set of files into the tag directory and commit

svn copy trunk tags/dlmf-1-0-2
svn commit -m "created dlmf-1-0-2 tag" tags

# Tagging a revision "after the fact" is like
```

```
svn copy -r <rev> http://<repo>/ http://<repo>/tags/<tag> -m <commit-comment>

# To see all tags:
svn list http://<repo>/tags/
# to show the tag message:
svn log --limit 1 http://<repo>/tags/<tag>

#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
# H. To setup a DLMF Server (including demo server on laptop)
#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
# The short form is:
#   (1) install java
#   (2) install tomcat
#   (3) drop a dlmf.war into the Right Place.
#   That's it.
# A slightly more sophisticated/secure installation has apache
# proxy for tomcat (which runs "behind" it)
#   (4) install apache
#   (5) configure apache to proxy serve:
# See linux install, below for details.
#
# More detailed explanation for specific platforms follows,
# and assumes you've built an appropriate war file (see above),
# or found one at
#   dlmf-dev.nist.gov:/local/tomcat5/dlmf-YYYYMMDD.war
# or
#   orion.cam.nist.gov:/local/dlmf/archive/dlmf-YYYYMMDD.war

#======================================================================
# Linux
#======================================================================
# On linux, it is best just to install tomcat using the standard package
# manager; It will install whatever dependencies, like java, that are needed.
# (note: on Centos, you may have to say tomcat5 in the following)
# On RPM based systems, use yum:

sudo yum install tomcat

# Now copy the war file to tomcat's webapp directory:

sudo cp dlmf-YYYYMMDD.war /var/lib/tomcat/webapps/dlmf.war

# After a few minutes, it should be available at

http://localhost:8080/dlmf

#----------------------------------------------------------------------
# Running apache as proxy for tomcat
#----------------------------------------------------------------------
# It may be worth installing httpd and serving dlmf through it.
# (certainly more secure on a public server)
```

```
sudo yum install httpd

# Create a configuration file, say /etc/httpd/conf.d/dlmf-tomcat.conf,
# containing :

ProxyRequests Off
ProxyPass /dlmf/ http://localhost:8080/dlmf/
ProxyPassReverse /dlmf/ http://localhost:8080/dlmf/

# Now, you should see dlmf at:

http://localhost/dlmf/

#======================================================================
# Mac
#======================================================================
# Offhand, I don't know actually, but it must be similar to
# the Linux; maybe there's a macport of tomcat?

#======================================================================
# Windows
#======================================================================
# Get JAVA:
#   Unless you've already got at least a Java 6 aka 1.6 version;
#   Go to
#       http://java.com/en/download/manual.jsp
#   Download & install the latest current Windows version.
#
#   If they give lots of choices ("beans", development kits,
#   "enterprise edition", etc.), a JRE version is fine.
#   (unless you plan to do your own development)
#
# Get TOMCAT:
#   Go to
#       http://tomcat.apache.org/
#   and choose the latest Tomcat 6.x.x version
#   (we haven't yet tested Tomcat 7)
#   Download and install an appropriate version from
#       "Binary Distributions/Core"
#
#   When running the installer, it may ask
#   which java to use: make sure it uses a nice
#   fresh one, if you installed one above.
#
#   IMPORTANT: You'll get some sort of tomcat manager
#   application. Run this and find "Tomcat Properties"
#   (or maybe the application _is_ Tomcat properties..)
#   You'll get a window with several tabs.
#   Under "Java" tab, find a box called "Java Options"
#   Click in that box and go to the end.
#   Add a new line that contains:
#      -Dfile.encoding=UTF-8
#    and save or whatever you need to do.

# Install dlmf.war
#   (1) Run the tomcat manager/tomcat properties and
#      make sure tomcat is stopped (click "Stop").
#   (2) Find the directory where tomcat got installed
#        <wherever>/apache-tomcat-6.x.x/webapps
#   (3) If you've installed dlmf before,
#       delete dlmf.war and any directory dlmf in webapps
#   (4) copy dlmf.war to the webapps.
#   (5) Run the tomcat manager/tomcat properties and
#      make start tomcat (click "Start").
#
#  Then, point your browser at
#     http://localhost:8080/dlmf/
#
#  and hopefully you're there, and everthing works!
#  Enjoy!

#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
# I. Other Esoterica...
#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
# A reminder of where to find the needed java jar files.
# The following jar files are used in the java runtime and need
# to be placed in
#
#      dlmf/web/WEB-INF/lib
#
#  === LUCENE: Search engine ===
#  Download lucene-java from http://lucene.apache.org/, untar
#  Find the appropriate lucene*.jar, EG.
#    lucene-core-1.9.1.jar
#
#  === XERCES: XML Parser ===
#  I've explicitly included xerce's (the xml parser) jars so that we have
#  the catalog resolver available (apparently not incldued with tomcat)

#  Download from Xerces-J-bin from http://xerces.apache.org/, untar
#  Find:
#    xercesImpl.jar
#    xml-apis.jar
#    resolver.jar

#  === XALAN: XSLT Transformer ===
#  I've explicitly included xalan's (the xslt engine) jars so that
#  extension functions are available.

#  Download Xalan-J-bin from http://xalan.apache.org/, untar
#  Find:
#    serializer.jar
#    xalan.jar
#    xsltc.jar
```