A User's Manual for GNU Emacs' Web-mode

Mark Motl & Bart Childs
Department of Computer Science
TEXas A&M University

Revision 2.00, April 17, 1993

This document is a companion to `web-mode.el`, a collection of GNU Emacs Lisp functions designed to customize GNU Emacs so that it is sensitive to a `WEB` document. Appendix A contains lists of the interactive functions in two formats. Most of these have convenient key-bindings. **It is assumed** that the reader is already familiar with the `WEB` style of literate programming, the general format of a `WEB` source, and the rules of `WEB`, `FWEB`, `CWEB`, or a similar one.

The intent in the design of `web-mode` is sensitivity to `WEB` independent of the high level language. Thus, it works with Knuth's original `WEB` with Pascal, `CWEB`, `FWEB`, and similar ones. The basic Emacs functionality is maintained along with showing matched delimiters and continuing indentation.

The functions that `web-mode` provides include: the indices of variables and sections provided by `WEAVE`, outline editing, navigation by chapter and section, preloading the search string when using indices, automatic completition in some places, ...

Throughout this document, I will use the symbology used by Emacs. For example, the `find-file` command can be entered two different ways

`M-x find-file` `C-x C-f`

The first way of entering the command is to actually type the command as shown above. Press the `META` key (on some terminals this may be the `ESC` key); next press the `x` key; finally, enter the characters `find-file` followed by a carriage return. An alternate way of issuing this command is to use its associated keybinding as shown in the box to the right of the command above. In this keybinding `C-` refers to the `CTRL` key. Press and hold down the `CTRL` key and then press the `x` key followed by holding down the `CTRL` key and then pressing the `f` key. For all commands mentioned throughout this manual, both forms of invoking the command will be shown if they exist. Some commands, such as

`M-x what-line`

do not have a keybinding and hence must be entered by typing in the command name. All commands that are `WEB` extensions have been prefixed with "`web-`". (The outline mode commands violate this.)

INSTALLATION

The installation instructions are really quite simple. To simplify this manual we will assume that these instructions are followed. The files `.emacs`, `limbo.sty`, and `limbo.material` are attached to the source of this TEX file for ease in distribution. (This began with release 1.6.) Of course these are the versions normally used at Texas A&M and should be adapted to your site(s).

1. Select a directory for the home of these files which can be shared by all users, we use `/usr/local/lib/web`.

   Extract `web-mode.el`, `limbo.material`, and `limbo.sty` from the source of this manual (after the `\bye`) and place them in that directory.

2. Extract `.emacs` from the same source and place it in **your home** directory.

   Edit the `.emacs` file to reflect the name of the directory. If it did not change, then you don't have to edit it. Appendix E is a more detailed description of this file and changes you can make.

3. Issue the command:
   `M-x byte-compile-file`

   The user will be prompted for the name of the file. Respond with `web-mode.el`. This will create a new file named `web-mode.elc`.

   Now, make sure that everybody can read and execute the files you have in this directory. Everybody with the same `.emacs` file in their home directory will automatically be able to use `web-mode` with their files ending in `.web` or `.w`

4. You may have to make some changes as detailed in Appendix E.

CAVEAT

Some of these functions may seem to be trivial and slow at first. We believe they will pay big dividends for larger `WEB`s.

Getting Started

The purpose of `web-mode` is to assist in creating and editing `WEB`s.

Creating a New `WEB`

If a `WEB` is being created, Emacs will need to be started with an empty file. For example, if the name of the new `WEB` is `my.web`, you would enter the UNIX command

```
% emacs my.web
```

The function `web-mode` will prompt the user for the name of the file that contains the limbo portion of the `WEB`. To accept the default file (`limbo.material`), the user need only press the return key. If, on the other hand, the user wishes to enter the name of another file, that can be done by answering with another file name. If the default is selected, the contents of the file `limbo.material` are inserted.

The user is prompted for the title of the `WEB`. This is not intended to be the name of the file, but a better name like, say the title of a book about the program being written.

The command invoking the editor showed an assumption that we feel should be honored. We recommend that the names of `WEB`'s end with `.web`. We also allow the use of the extension `.w` because that is how `CWEB` is distributed.

We also recommend that included files (*assuming you are using* `@i` *commands available in* `CWEB`*and* `FWEB`) **not** end in `.web`. We recommend endings like `.hweb`. It just seems more *literate!*

Editing a `WEB` Created Without `web-mode`

If you wish to use `web-mode` on an existing `WEB`, say `primes.web`, you would begin by invoking Emacs with the UNIX command

```
% emacs primes.web
```

When the function `web-mode` is executed, it will perform some initialization of various data structures. A list of the section names is collected. If `web-mode` detects that a section name has been used but never defined, a "stub" section is inserted at the **end of the chapter** in which it was first used. A stub section is like:

```
@␣@^Stub@>
@<Name of undefined section@>=
```

`Web-mode` also reads the `CHange` file that accompanies the `WEB` currently being edited. If there is no `CHange` file, one is created; however, this file is "saved" only if it is modified.

The WEB Is Like A Book

The user should understand that we are using a 'book model' for a WEB. The basic element of a WEB is a section which is roughly parallel to a paragraph in a book. The 'major sections' or group of sections corresponds to a chapter in a book. Chapter titles will be in the table of contents. We have an extension of the book model in that many sections will have 'names' that will appear in an index. Technical books may have subjects of paragraphs in an index.

Movement Within the WEB

The user can goto any section or chapter at any time by issuing either:

`M-x web-goto-section`	`C-c g s`
`M-x web-goto-chapter`	`C-c g c`

respectively. The user is then prompted for the section or chapter number. If a nonpositive number or a number greater than the number of sections or chapters in the WEB is entered, an appropriate error message is displayed in the minibuffer. If a valid section or chapter number is entered, point moves to the beginning of that section or chapter in the WEB. The command can also be entered with a numeric argument prefix which some experienced Emacs users may wish to use.

If the change file has an entry for the destination section, a message is printed in the minibuffer stating that the destination section has been edited.

The user can also goto the next or previous section or chapter by entering the commands:

`M-x web-next-section`	`C-c n s`
`M-x web-previous-section`	`C-c p s`
`M-x web-next-chapter`	`C-c n c`
`M-x web-previous-chapter`	`C-c p c`

If there is no next or previous section (or chapter), an appropriate message is displayed in the minibuffer; otherwise, point moves to the beginning of the next or previous section (or chapter), appropriately.

The user can determine the section or chapter (number) that point is positioned within by the commands

`M-x web-which-section`	`C-c w s`
`M-x web-which-chapter`	`C-c w c`

Movement Among the Buffers

Several commands facilitate movement within the WEB document. The user can switch to the buffer containing the CHange file by issuing the command

`M-x web-goto-buffer-change-file` `C-c b c`

Similarly, the user can switch to the buffer containing the WEB file at any time with the command

`M-x web-goto-buffer-web-file` `C-c b w`

If the WEB being edited is a CWEB, the user can switch to any of the include files with the command

`M-x web-goto-buffer-include-file` `C-c b i`

This command takes a numeric argument. For example, giving the function an argument of 1 would cause the buffer containing the contents of the first include file command (@i) encountered in the CWEB to become the current buffer.

Unfortunately, web-mode can be confusing as to which buffer you want. In particular, if there are multiple includes, how does one know which one to go to? Fortunately a solution is provided via the command:

`M-x buffer-menu` `C-x C-b`

When invoked, this command opens a window containing a list comprised of the following files: the WEB file, the CHange file, and the include files. The user types the command '`C-x o`' to change the point to the window with the list of files. Then, the user uses the next (and previous) line command to place the point of the desired buffer (file). The user enters an '`f`' to find that file. The user may wish to execute the command '`C-x 1`' to return to a single window.

If you know Emacs, you know that several other buffers will be visible as well. The information kept for using the index of variables and index of section names are buffers too.

Inserting a Section Name

Whenever the < key is pressed in the buffer containing the WEB document, the function web-insert-section-name is invoked. This function must determine whether this is the beginning of a section name or not. The function examines the preceding characters. If an odd number of @'s immediately precede the <, then it is the beginning of a section name and web-mode will automatically invoke the function web-options-for-section-name-insertion; otherwise, the < is inserted as the user continues typing.

If the function determines that it is the beginning of a section name, the user is switched to a buffer with contents like those shown in Figure 1. The user must then select one of the options given. Figure 1 is based on primes.web, a n example in [1].

```
Option  Action
------  ----------------------------------------
   A    Abort
   C    Create a New Section Name
   L    List all Section Names Beginning with Letter
   N    Next Screen
   P    Previous Screen
   S    Select Existing Section Name
-------------------------------------------------------------------------
Name -- for use with Selection based upon completion
  Fill table |p| with the first |m| prime numbers
  Give to |j_prime| the meaning: |j|~is a prime number
  If |p[n]| is a factor of~|j|, set |j_prime:=false|
  Increase |j| until it is the next prime number
  Initialize the data structures
  Other constants of the program
  Output a line of answers
  Output a page of answers
  Print table |p|
  Print the first |m| prime numbers
  Program to print the first thousand prime numbers
                        Displaying 11 Section Names of 14
--**-Emacs: *Section Names*        (Web)----All----------------------------
```

Figure 1. Section names for Knuth's primes.web

A.OC If this *abort* option is selected, the user is switched back to the buffer containing the WEB document and the two characters preceding point (@<) are deleted.

The remaining options involve the list of section names. The section names are displayed below the menu choices. The section names are displayed in alphabetical order and are retrieved from the list that was created when web-mode was invoked

at the beginning of the editing session. Also, additional names are inserted in the list as they are created in an editing session.

`C.` If the option is chosen, the user is prompted for the name of the new section. Once entered, the user is switched back to the buffer containing the `WEB` document, the new section name is inserted, and the ending section delimiter (either `@>` or `@>=`) is determined automatically and inserted. A check against the existing section names is conducted to ensure that the entered section name is indeed new. If the section name is new, a determination of whether a stub section needs to be appended is made and the new section name is added to the list of section names.

`L.` This causes all section names that begin with a particular letter to be displayed. The user is prompted for the letter. If no section names begin with the desired letter, an appropriate message is displayed in the minibuffer.

`N.` This causes the next screen of section names to be displayed. For example, if the screen is capable of displaying 21 section names and currently section names 1 through 21 are displayed, choosing `N` would cause section names 22 through 42 to be displayed.

`P.` This causes the previous screen of section names to be displayed. For example, if the screen is capable of displaying 21 section names and currently section names 30 through 50 are displayed, choosing `P` would cause section names 9 through 29 to be displayed.

`S.` **This option has changed** from previous versions. It now expects to do selection by a **completion** scheme that is popular with Emacs users. The `L`, `N`, and `P` options can be helpful. It is not required that the name being selected is visible on the screen. (Previous versions were based upon the number within a list.)

This might seem like a *severe interruption* of the coding process. However, getting the "named sections" spelled and spaced correctly is *critical.*

Viewing the Index, Section Names, and Chapter Titles

`Web-mode` allows a programmer to view the index as created by `WEAVE`. A user can also display the list of section names that is maintained by `web-mode`. It is also possible to view the names of the chapters of the `WEB` and the section number where each chapter begins.

Viewing the Index

The user can view the contents of the index created by `WEAVE` by issuing the command

`M-x web-view-index` `C-c v i`

The user is first asked whether the `WEB` is a Pascal, `CWEB`, or `FWEB`. After the language has been determined, the function creates an asynchronous process which invokes the `WEAVE` processor on the `WEB` in the current buffer. Once this process has completed, the contents of `jobname.idx` are inserted into another buffer, see Figure 2. Some reformatting is done for a better presentation. The user can traverse the list of section numbers for a particular index entry by positioning the point anywhere on the line(s) where the index entry of interest is located and issuing the commands:

`M-x web-next-index` `C-c n i`

`M-x web-previous-index` `C-c p i`

These commands can be issued from the buffer where the index is displayed or they can be issued from the buffer that contains the `WEB` document. If the commands are issued from the former, the user is switched back to the buffer that contains the `WEB` document, and point is positioned at the beginning of the section where the most recently selected index entry was next or previously referenced from point's current position.

Important The `web-view-index` has calls to the Emacs' function `call-process`. The first argument to this function is the name of the program that is to be run as a separate process. If the program names of the `WEAVE` processors differ on your system, elisp variables need to be reset accordingly in the file `.emacs`.

At Texas A&M we have modified `WEAVE` so that it writes the index to the file `jobname.idx` and the list of section names to the file `jobname.scn`. (The changes that must be made to `WEAVE` to accomplish this are contained in Appendix C.)

```
Bertrand, Joseph, postulate: 21.
boolean: 15.
c: 7.
cc: 5, 7, 8, 10.
Dijkstra, Edsger: 1, 15.
Eratosthenes, sieve of: 24.
false: 13, 26.
integer: 4, 7, 12, 17, 24.
j: 12.
j_prime: 13, 14, 15, 22, 26.
k: 12.
Knuth, Donald E.: 15.
m: 2.
mult: 24, 25, 26.
n: 23.
new_line: 6*, 9, 10.
new_page: 6*, 9.
ord: 17, 18, 19, 20, 21, 22, 23, 24, 25.
ord_max: 17, 19, 23, 24.
output: 2, 6*.
output format: 5, 9.
p: 4.
--**-Emacs: *INDEX for primes.web*    (Web)----Top--------------------------
```

Figure 2. Index for Knuth's `primes.web`

Viewing the Section Names

The user can also view the contents of the list of section names that is maintained by `web-mode`. When the command

`M-x web-view-section-names-list` `C-c v s`

is issued, the list is displayed in another buffer. Some reformatting is necessary. The user is free to peruse the list as much as he likes. The results of issuing this command on Knuth's `primes.web` is shown in Figure 3.

The commands

`M-x web-next-define` `C-c n d`

`M-x web-next-use` `C-c n u`

`M-x web-previous-define` `C-c p d`

`M-x web-previous-use` `C-c p u`

can be invoked from the buffer where the section names are displayed or they can be invoked from the buffer containing the `WEB` document. If they are invoked from the former, the user selects a section name by positioning the point anywhere on the line(s) where the section name of interest is located. Once a section name has been selected, the user is switched back to the buffer containing the `WEB` document

```
Fill table |p| with the first |m| prime numbers (11) (3)
Give to |j_prime| the meaning: |j|~is a prime number (22) (14)
If |p[n]| is a factor of~|j|, set |j_prime:=false| (26) (22)
Increase |j| until it is the next prime number (14) (11)
Initialize the data structures (16 18) (11)
Other constants of the program (5 19) (2)
Output a line of answers (10) (9)
Output a page of answers (9) (8)
Print table |p| (8) (3)
Print the first |m| prime numbers (3) (2)
Program to print the first thousand prime numbers (2) (1)
Update variables that depend on~|j| (20) (14)
Update variables that depend on~|ord| (21 25) (20)
Variables of the program (4 7 12 15 17 23 24) (2)
--**-Emacs: *Section Name (Defined In) (Used In)*   (Web)--All--------------
```

Figure 3. Section names list for Knuth's `primes.web`

at the section where the most recently selected section name was next/previously defined or used, whichever is appropriate, from point's current position.

While in the `WEB` document, the "Defined In" and "Used In" lists can be traversed with the commands

`M-x web-next-define` `C-c n d`

which positions point at the beginning of the section where the most recently selected section name was next defined,

`M-x web-previous-define` `C-c p d`

which positions point at the beginning of the section where the most recently selected section name was previously defined,

`M-x web-next-use` `C-c n u`

which positions point at the beginning of the section where the most recently selected section name was next used, and

`M-x web-previous-use` `C-c p u`

which positions point at the beginning of the section where the most recently selected section name was previously used.

It should be noted that these commands prevent the user from running off either end of either list.

`Web-mode` maintains a list of the form

(("section-name-1" (Defined In Sections) (Used In Sections))
("section-name-2" (Defined In Sections) (Used In Sections))
...
("section-name-n" (Defined In Sections) (Used In Sections)))

This list is collected automatically by `web-mode` and dynamically maintained. If the user feels that this list is in error at any time during an editing session, the list

can be recollected with the command

```
M-x web-collect-section-names
```

Viewing the Chapter Titles

The user can also view a list of the titles of the chapters in the WEB document and the section number where each chapter begins. The command

```
M-x web-view-chapter-titles-list                    C-c v c
```

collects the names of the chapters along with their section numbers and displays them in another buffer. The user is switched over to this buffer. The result of issuing this command on Knuth's `primes.web` is shown in Figure 4.

```
Chapter# --  Chapter Title
   1     Printing primes: An example of \web
   2     Plan of the program
   3     The output phase
   4     Generating the primes
   5     The inner loop
   6     Index
--**-Emacs: *Chapter Titles*    (Web)----All--------------------------------
```

Figure 4. Chapter titles for Knuth's `primes.web`

While viewing this list, the user can issue the command '`C-c g s`' and respond with the number of the chapter indicated above. The '`C-c n s`' and '`C-c p s`' commands are quite handy for moving to the next and previous chapters too.

Support for Change Files

`Web-mode` also provides support for `change` files. The command

`M-x web-view-edited-sections-list` `C-c v e`

can be invoked in either the buffer containing the `WEB` document or the buffer containing the `change` file. It displays, in a separate window, a list of the numbers of sections that have been edited.

The command

`M-x web-edit-section` `C-c e s`

will insert the contents of the section that point is positioned in when the command is invoked in the `change` file in its proper position. The entire section's contents are copied twice—once between `@x @y` and once between `@y @z`. The user is positioned at the first line after the `@y`. This command is only available in the buffer containing the `WEB` document.

The command

`M-x web-which-edited-section` `C-c w e`

can only be invoked in the buffer containing the `change` file. This function displays a descriptive message informing the user of what section in the `WEB` the change that point is positioned in corresponds to.

Movement within the `change` file is accomplished with the commands:

`M-x web-goto-edited-section` `C-c g e`

The user is prompted for a numeric argument. Point is then positioned at the beginning of the change in the `change` buffer that corresponds to the section number given as an argument.

`M-x web-next-edited-section` `C-c n e`

This command positions point at the beginning of the next change in the `change` buffer:

`M-x web-previous-edited-section` `C-c p e`

This command positions point at the beginning of the previous change in the `change` buffer.

The command

`M-x web-count-edited-sections` `C-c # e`

will display the number of sections that have been edited.

Outline Mode

An outline-mode has been included based on other Emacs modes. The changes necessary were to decide what the strings are that define the various levels for the outline.

These operations hide all information execpt the headings and then reverse the operation. This applies to the entire buffer.

`M-x hide-body` `C-c h b`

`M-x show-all` `C-c s a`

Once you have put the buffer into outline mode, you can navigate in the buffer based on these headings.

`M-x outline-next-visible-heading` `C-c n h`

`M-x outline-previous-visible-heading` `C-c p h`

The outline form can have headings at several levels visible. These commands allow you to navigate based upon traveling at the same level of heading (skips subheadings) and to move from a subheading to a higher level.

`M-x outline-forward-same-level` `C-c f l`

`M-x outline-backward-same-level` `C-c b l`

`M-x outline-up-heading` `C-c u h`

The immediately following body part can be made invisible or visible with these commands.

`M-x hide-entry` `C-c h e`

`M-x show-entry` `C-c s e`

The user can make parts of the body and subheadings invisible or visible with these commands.

`M-x hide-subtree` `C-c h s`

`M-x show-subtree` `C-c s s`

The user can show children thereby making direct subheadings visible. It does not affect the visibility of the body or subheadings two (2)or more levels down. This command may be preceeded with a `C-u N` command to furnish an argument. If omitted, the value of one (1) is used.

`M-x show-children` `C-c s c`

To make the body under heading and under its subheadings invisible while the subheadings remain visible:

`M-x hide-leaves` `C-c h l`

To make all subheadings at all levels visible:

`M-x show-branches` `C-c s b`

Miscellaneous Functions

The function
`M-x web-count-sections` `C-c # s`
displays in the minibuffer the total number of sections in the `WEB` document that is currently being edited.

The function
`M-x web-count-chapters` `C-c # c`
displays in the minibuffer the total number of chapters in the `WEB` document that is currently being edited.

The command
`M-x web-delimiter-match-check` `C-c d m`
looks at the entire `WEB` document section by section. It determines if the delimiters surrounding section names, namely, `@<` and `@>`, are matched. If the delimiters do not match properly, a descriptive message is printed in a temporary buffer.

`Web-mode` also makes it easier to enter index entries. When the function
`M-x web-insert-index-entry` `C-c i i`
is issued, the user is prompted for two things:

1. The type of font to be used (Roman, typewriter, or user-defined). This determines the opening delimiter to the index entry (either `@^`, `@.`, or `@:`) and
2. The text of the index entry.

The complete index entry is then placed on a new line.

The command
`M-x web-rename-section` `C-c r s`
allows the user to rename a section. Point should be positioned on the name that the user wishes to rename. The user is prompted for the new name. The user is then given the option of renaming a single occurrence, all occurrences, or is queried if each of the remaining occurrences should be renamed.

There are several functions in `web-mode` that are invoked automatically whenever a particular key is pressed. These functions include:
`web-AT-sign-processing`
which is invoked whenever the `@` key is pressed. The function checks to see if the `@` is at the beginning of a line. If it is, the next character is read. If this next character is either a space, newline, tab, or asterisk, then the `@` in combination with the next character constitutes the beginning of a new section.

Another useful function that is invoked automatically is

`web-newline`

which is invoked whenever the newline character is pressed. This function positions point so that the next line has the same indentation as the current line that is being terminated.

One keybinding that Emacs' users are familiar with has been rebound to another function. The keybinding `C-x C-c` has been bound to the `web-mode` function

`M-x web-mode-save-buffers-kill-emacs` `C-c x c`

Before this function makes a call to Emacs' `save-buffers-kill-emacs`, it completes the journal file that is maintained by `web-mode`. This journal file keeps track of what file is being edited, who is editing it, when the editing began, when the editing stopped, and a count of the number of times each of `web-mode`'s interactive functions were used.

By default, `web-mode` maintains a journal file. This file contains the user's name, the name of the file being edited, the start and stop times of the editing session, total time in `web-mode`, and a count of each of the interactive `web-mode` functions invoked during the editing session. The collection of this file is governed by the Boolean variable `web-journal-on` which is set to `t` (true). This can be turned off by changing a line in `.emacs` to `(setq web-journal-on nil)`.

An Example

Let's assume that you are creating `main.web` which will input files `sub_1.hweb` and `sub_2.hweb`. After entering the statement `@i sub_1.hweb` we suggest that you `C-xC-f` and respond with the `sub_1.hweb`. Enter one blank line and `C-xb` followed by an enter and you will be back into the `WEB` file. Entering `M-x web-mode-reset` will ensure that you can now communicate among the files in the preferred manner.

This same process is repeated to create the other included file. Of course, if they already existed then you did not need to visit them to create them. **Again,** we consider it slightly illiterate if these files have an extension of `.web`!

Try it out on `NewtonC.web`, NewtonF.web, `PS_Quasi.web` and the sources of the different `WEB` systems.

This page (and 3) should be a bit longer or another document of examples should exist.

A Final Note

Any suggestions for enhancements, problems with the existing functions, or criticisms can be submitted to

- Bart Childs (`bart@cs.tamu.edu`).

Mark Motl is not available through the networks at this time. The use of the pronoun 'I' obviously should be 'we.'

References

1. Donald E. Knuth, "Literate Programming," *The Computer Journal*, vol. 27, no. 2, May 1984, pp. 97–111.
2. Bill Lewis, Dan LaLiberte, and the GNU Manual Group, *The GNU Emacs Lisp Reference Manual: Emacs Version 18 for UNIX Users*, Edition 1.01, Cambridge, MA: Free Software Foundation, Apr. 1990.
3. Richard Stallman, *GNU Emacs Manual*, Fifth Edition, Version 18 for UNIX Users, Cambridge, MA: Free Software Foundation, Oct. 1986.

Supplementary Sources

1. Adrian Avenarius and Siegfried Oppermann, "FWEB: A Literate Programming System for Fortran8x," *SIGPLAN Notices,* vol. 25, no. 1, Jan. 1990, pp. 52–58.
2. Jon Bentley, "Programming Pearls: Literate Programming," *Communications of the ACM*, vol. 29, no. 5, May 1986, pp. 364–369.
3. Jon Bentley and David Gries, "Programming Pearls: Abstract Data Types," *Communications of the ACM*, vol. 30, no. 4, Apr. 1987, pp. 284–290.
4. Jon Bentley, Donald E. Knuth, and Doug McIlroy, "Programming Pearls: A Literate Program," *Communications of the ACM*, vol. 29, no. 6, Jun. 1986, pp. 471–483.
5. Marcus E. Brown, "An Interactive Environment for Literate Programming," Ph.D. Dissertation, Dept. of Computer Science, Texas A&M University, College Station, TX, Aug. 1988.
6. Marcus E. Brown and Bart Childs, "An Interactive Environment for Literate Programming," *Structured Programming*, vol. 11, no. 1, Jan. 1990, pp. 11–25.
7. Peter J. Denning, "Announcing Literate Programming," *Communications of the ACM*, vol. 30, no. 7, Jul. 1987, p. 593.
8. Klaus Guntermann and Joachim Schrod, "`WEB` Adapted to C," *TUGboat*, vol. 7, no. 3, Oct. 1986, pp. 134–137.
9. Eric Hamilton, "Literate Programming: Expanding Generalized Regular Expressions," *Communications of the ACM*, vol. 31, no. 12, Dec. 1988, pp. 1376–1385.

10. David R. Hanson, "Literate Programming: Printing Common Words," *Communications of the ACM*, vol. 30, no. 7, Jul. 1987, pp. 594–599.
11. Michael A. Jackson, "Literate Programming: Processing Transactions," *Communications of the ACM*, vol. 30, no. 12, Dec. 1987, pp. 1000–1010.
12. Donald E. Knuth, METAFONT*: The Program*, vol. D of *Computers & Typesetting*, Reading, MA: Addison-Wesley Publishing Co., 1986.
13. Donald E. Knuth, *TEX: The Program*, vol. B of *Computers & Typesetting*, Reading, MA: Addison-Wesley Publishing Co., 1986.
14. Donald E. Knuth, "The WEB System of Structured Documentation," Stanford Computer Science Report STAN-CS-980, Dept. of Computer Science, Stanford University, Stanford, CA, Sep. 1983.
15. John A. Krommes, "The WEB System of Structured Software Design and Documentation for Fortran, Ratfor, and C," Technical Report, Princeton University, Princeton, NJ, Nov. 1989. Available by anonymous ftp from lyman.pppl.gov in directory /pub/fweb.
16. Silvio Levy, "WEB Adapted to C, Another Approach," *TUGboat*, vol. 8, no. 1, Apr. 1987, pp. 12–13.
17. Donald C. Lindsay, "Literate Programming: A File Difference Program," *Communications of the ACM*, vol. 32, no. 6, Jun. 1989, pp. 740–752.
18. Charles Lins, "A First Look at Literate Programming," *Structured Programming*, vol. 10, no. 1, 1989, pp. 60–62.
19. Charles Lins, "An Introduction to Literate Programming," *Structured Programming*, vol. 10, no. 2, 1989, pp. 107–112.
20. Norman Ramsey, "Literate Programming: Weaving a Language-Independent WEB," *Communications of the ACM*, vol. 32, no. 9, Sep. 1989, pp. 1051–1055.
21. E. W. Sewell, "How to MANGLE Your Software: The WEB System for Modula-2," *TUGboat*, vol. 8, no. 2, Jul. 1987, pp. 118–122.
22. E. W. Sewell, *Weaving a Program: Literate Programming in WEB*, New York, NY: Van Nostrand Reinhold, 1989.
23. Richard M. Stallman, "EMACS: The Extensible, Customizable, Self-Documenting Display Editor," in *Interactive Programming Environments*, David R. Barstow, Howard E. Shroke, and Erik Sandewall, Eds., New York, NY: McGraw-Hill Book Co., 1984, pp. 300–325.
24. Harold Thimbleby, "Experiences of 'Literate Programming' using cweb (a variant of Knuth's WEB)," *The Computer Journal*, vol. 29, no. 3, Jun. 1986, pp. 201–211.
25. Sho-Huan Tung, "A Structured Method for Literate Programming," *Structured Programming*, vol. 10, no. 2, 1989, pp. 113–120.
26. Christopher J. Van Wyk, "Literate Programming: An Assessment," *Communications of the ACM*, vol. 33, no. 3, Mar. 1990, pp. 361 and 365.

Appendix A

Assorted Information and Tables

With the exception of the function `web-mode-save-buffers-kill-emacs` all keybindings have a similar form that consists of three parts. Each command is prefaced with `C-c`. The rest of the keybinding is the first letter of two following words in the function name.

Alphabetized Listing of `Web-mode` Commands

Command	Key Binding
web-count-edited-sections	C-c # e
web-count-sections	C-c # s
web-count-chapters	C-c # c
web-delimiter-match-check	C-c d m
web-edit-section	C-c e s
web-goto-buffer-change-file	C-b b c
web-goto-buffer-include-file	C-b b i
web-goto-buffer-web-file	C-b b w
web-goto-chapter	C-c g c
web-goto-edited-section	C-c g e
web-goto-section	C-c g s
web-goto-section-named	C-c g n
web-insert-index-entry	C-c i i
web-mode-save-buffers-kill-emacs	C-x C-c
web-next-chapter	C-c n c
web-next-define	C-c n d
web-next-edited-section	C-c n s
web-next-index	C-c n i
web-next-section	C-c n s
web-next-use	C-c n u
web-previous-chapter	C-c p c
web-previous-edited-section	C-c p e
web-previous-define	C-c p d
web-previous-index	C-c p i
web-previous-section	C-c p s
web-previous-use	C-c p u
web-rename-section	C-c r s
web-view-chapter-titles-list	C-c v c
web-view-edited-sections-list	C-c v e
web-view-index	C-c v i
web-view-section-names-list	C-c v s
web-which-chapter	C-c w c
web-which-edited-section	C-c w e
web-which-section	C-c w s

Listing of `Web-mode` Commands by Functionality

Functionality	Command	Key Binding
Movement Among Buffers (Files)	web-goto-buffer-change-file	C-c b c
	web-goto-buffer-include-file	C-c b i
	web-goto-buffer-web-file	C-c b w
Movement Among Sections	web-goto-section-named	C-c g n
	web-goto-section	C-c g s
	web-next-section	C-c n s
	web-previous-section	C-c p s
Interactive Access to and Movement Among Chapters	web-goto-chapter	C-c g c
	web-next-chapter	C-c n c
	web-previous-chapter	C-c p c
	web-view-chapter-titles-list	C-c v c
Interactive Access to Index	web-next-index	C-c n i
	web-previous-index	C-c p i
	web-view-index	C-c v i
Interactive Access to Sections	web-next-define	C-c n d
	web-next-use	C-c n u
	web-previous-define	C-c p d
	web-previous-use	C-c p u
	web-view-section-names-list	C-c v s
Change File Editing & Movement	web-edit-section	C-c e s
	web-goto-edited-section	C-c g e
	web-next-edited-section	C-c n e
	web-previous-edited-section	C-c p e
`Web` Structure Information	web-count-edited-sections	C-c # e
	web-count-sections	C-c # s
	web-count-chapters	C-c # c
	web-delimiter-match-check	C-c d m
	web-view-edited-sections-list	C-c v e
	web-which-edited-section	C-c w e
	web-which-section	C-c w s
	web-which-chapter	C-c w c
Miscellaneous	web-insert-index-entry	C-c i i
	web-mode-save-buffers-kill-emacs	C-x C-c
	web-rename-section	C-c r s

Listing of `Web-mode` Commands by Functionality

Functionality	**Command**	**Key Binding**
Outlining the Whole Buffer	`hide-body`	`C-c h b`
	`show-all`	`C-c s a`
Move By Visible Heading	`outline-next-visible-heading`	`C-c n h`
	`outline-previous-visible-heading`	`C-c p h`
Movement at the Same Level	`outline-forward-same-level`	`C-c f l`
	`outline-backward-same-level`	`C-c b l`
	`outline-up-heading`	`C-c u h`
Entry Actions	`hide-entry`	`C-c h e`
	`show-entry`	`C-c s e`
Subtree Subtree	`hide-subtree`	`C-c h s`
	`show-subtree`	`C-c s s`
Miscellaneous Outlining	`show-children`	`C-c s c`
	`hide-leaves`	`C-c h l`
	`show-branches`	`C-c s b`

These commands need to be reworked to agree with the `web-` prefix and to ensure a little more consistency.

Appendix B

Basic Information Like Variables, etc.

Previous versions of this manual included a list of global variables. The reason for this list is to help debugging in case of errors. This is cumbersome to try to keep updated as we begin to have more users and contributors to the evolution of this package.

We have replaced this by some instructions as to how to generate a current list and how that is used. The list of global variables is easily generated by finding all lines with the string '`defvar`' on them. Using `grep` this is simply:

```
grep defvar web-mode.el > DefVar
```

On UNIX-like systems, the file `DefVar` will contain the list of variables with a little more information.

The documentation for the above can be viewed by issuing the command:

`M-x describe-variable` `C-h v`

At the prompt, the name of the variable of interest should be entered. The response will be the help text entered when the variable was declared and its current value.

Appendix C

Changes To The WEB Processors

One of the best things to do in software is to reuse effective and correct code. The use of the index is one of the best parts of the WEB system. It did not take long for us to decide that we saw no reason for Mark to write lots of elisp code to produce an index when it was already produced by the WEAVE processors. To facilitate the reuse of this a little more, it was decided to have `jobname.idx` and `jobname.scn` written separately. The first is used by `web-mode` while the second is not.

A companion file to this distribution is `changes.ch` which includes the changes necessary to the most popular WEAVE processors. Short comments about these contents follow.

In the original WEAVE, module 239 must be edited. The current version of this code is 4.????

Another alternative for users of the original WEAVE is available in the code. Look for the string "`Gragert`" if you wish to use this and not make these changes.

Version 2.9 and later versions of CWEB do not require changes.

At this writing, no code changes are required for FWEB. The current version of this code is 1.23. Edit `fweb.sty` to get the desired results with FWEAVE. It should look (in part) like this:

```
index.tex "#.idx"
modules.tex "#.scn"
contents.tex "#.cts"
```

Appendix D

Debugging, Hopefully

This will be a *pot pourri* of notes that may help some unsuspecting user solve some problems that occur in the use of `web-mode`. The first item came about from personal experiences of Bart Childs' helping such users. (There used to be more in this list but they have disappeared due to improvements in `web-mode`. The users are requested to send in more ...)

1. Watching the minibuffer gives you an idea as to what `web-mode` is doing as it builds its pointers and lists. It is not uncommon for `WEB`s written without `web-mode` to have problems when used with `web-mode` for the first time. In at least one case, it is still not obvious as to why we had the problem.

   The solution of this case might be a help to other users. The starting of `web-mode` was nearing an end because it gave a message about collecting "web index entries." Then, the terrible message
   `File mode specification error: (wrong-type-argument stringp nil)`
   which seemed totally uncalled for.

   A method of solution. If the `WEB` has a good number of user supplied index entries, then you can find a pretty good bracket of the error easily. This is done by issuing the emacs command: `C-hv`. Emacs will prompt for the name of the elisp variable you want described. The answer is:
   `web-index-entries-list`
   and the last entry in it will be the last one processed. If all entries are entered only once, then the error will have happened between the point of the last entry in the list and the next entry that should have been added.

   I suggest adding a lot of entries like:

   ```
   @.BC 1 CB@>
   @.BC 2 CB@>
   ...
   ```

   say, after each paragraph. Repeating the `C-hv` iteration should quickly lead you to the offending item. Sometimes it is just a stray `@`?

2. Let us hear from you.

Appendix E

Some Elementary `.emacs`

This will be a brief discussion of the really necessary things and a few suggestions as to what goes in the file `.emacs`. Some of these may not be in exactly same order as in the file.

This was recommended by somebody, just do it.

```
(setq text-mode-hook 'turn-on-auto-fill)
```

These lines make the invocation of `web-mode` automatic if the name of the file ends with `.web`. The second line contains the name of the file where the *byte-compile*d form of `web-mode.el` is on your system.

```
(setq auto-mode-alist (cons '("\\.web\$" . web-mode) auto-mode-alist))
;; the next line also allows .w extensions
(setq auto-mode-alist (cons '("\\.w\$" . web-mode) auto-mode-alist))
(autoload 'web-mode "/usr/local/lib/web/web-mode.elc"
  "Major mode for editing WEB-based documents" t)
```

The next several show suggestions for the location of the *weave*s.

```
(setq web-call-program-cweave "/usr/local/bin/cweave")
(setq web-call-program-fweave "/usr/local/bin/fweave")
(setq web-call-program-weave "/usr/local/bin/weave")
```

The location of the default "limbo.material" file is shown. Individual users may create and use personalized limbo files by editing this line in `.emacs`

```
(setq web-limbo-material-location "/usr/local/lib/web/limbo.material")
```

These lines are included for users who don't like our ways. We find them to be convenient and recommend they be commented or just left out of your `.emacs` file.

```
(setq web-journal-on t)
(setq web-i-do-not-want-to-see-section-names t)
(setq web-i-do-not-want-stubs t)
```

Appendix F

Frequently Asked Questions, (FAQ)

Q: How do I get started?

A: After the `\bye` statement in the source of this document is three files. The first is `.emacs` and it should be installed in the user's root directory. There is some obvious editing that should be done to that file based upon the location of a few things. You should also put `limbo.sty`, `limbo.material` and **byte-compile** and put `web-mode.el` and `web-mode.elc` in appropriate places.

Q: If `web-mode` is properly installed, how do I get started?

A: Simply issue the command `?emacs start.web` where the `?` is an acknowledgement that not all emacs invocations start the same. The system will prompt you for acceptance of the use of the standard limbo material which you normally accept with (ENTER). You will then be prompted for a title. This should be a few words, say like a book title. Go read the manual and code, document, and be literate!

Q: What causes me to get a message like:
`"file-mode-specification-error (wrong-type-argument stringp nil)"`

A: That probably happens from a messed up index entry! Get out of the editor saving the web file and try these:

```
grep -e'@\.' file.web
grep -e'@\^' file.web
grep -e'@9'  file.web
```

or the combination of all:

```
grep -e'@[9\.\^]' file.web
```

If there is not one of the index entry beginnings without an ending, then you should probably email the file to `bart@cs.tamu.edu` or some other willing fool. Hopefully the experience of finding the answer will make it to this list. **Thanks** to Marcus Speh for starting such thoughts.

Appendix G

Glossary

buffer — A buffer is the basic editing unit. In Emacs, a user can have many buffers, however, only one can be edited at any one time.

change file — This file contains information that is used to override portions of the `WEB` file. Each "change" consists of repeating the lines from the `WEB` file that require modification and the new lines that are to replace the old ones. The `TANGLE` and `WEAVE` processors have the ability to scan both the `WEB` and change files so that the new lines replace the old lines at the appropriate places in the resulting high-level language and TEX codes.

chapter — We use chapter to mean a collection of sections. A chapter is begun by the `@*` character pair.

chapter title — The string of characters following the `@*` until the first period followed by white space (space, tab, new-line). The chapter title is presented in a bold font in the section and in the default font in the table of contents.

include file — Some `WEB` systems allow the inclusion of code through the use of `@i` commands. This is normally used in a slightly different manner than the C header files.

limbo file — A file named containing limbo material. The name is set in the `.emacs` file.

limbo material — A set of TEX material that aids in the complete presentation of a `WEB`. The distributed `limbo.material` includes source for a title, abstract, author, and other handy information.

minibuffer — The bottom line of the screen when using Emacs. This area is used to display arguments to commands, commands that are typed in from the keyboard, and messages to the user. For commands that require arguments, the minibuffer area can be used for prompting as well.

numeric argument — A number which is specified before a command to alter the effect of the command.

section — A section begins with the pair of symbols '`@*`' or '`@␣`.' A section ends when the next section begins or the end of the file, whichever comes first. If it begins with an '`@*`' it will be called a chapter.

section name — Code portions of sections which begin `@<Name of a section@>=` define the 'section name' by the strings between the `@<` and `@>` delimiters. Thus, section names are pseudocode descriptions of code function.

web file — contains the source of the `WEB`. The file consists of the limbo portion and the individual units that comprise the `WEB` called sections. A section consists of a commentary, definition, and code parts, any two of which can be empty.

Readers, please suggest other items for this!!!!!