Fountain Mode

GNU Emacs major mode for screenwriting in Fountain markup

Paul W. Rankin

The following people contributed to this documentation:
Paul W. Rankin, Kohl Sudduth

Overview

Fountain Modecombines the simplicity of Fountain syntax with the extensibility of Emacs. Fountain Modeis a major mode for Emacs, which is to say it is a program that runs inside Emacs — the extensible, customizable, free/libre text editor. It enables the writer to engage with all aspects of realizing a screenplay — story development, outlining, drafting, and rewriting.

To quickly get up to speed with Fountain syntax, memorize the rules for the six most used elements of the screenplay format: [Scene Headings], page 3, [Action], page 3, [Characters], page 3, [Dialogue], page 4, [Parentheticals], page 4, and [Transitions], page 4.

Then learn how to outline a script. See [Section Headings], page 2. This will be familiar for anyone who has used Markdown, as the syntax is the same. Sections allow you to easily show or hide and reorder large parts of a script.

There are additional Fountain syntax elements to allow for brainstorming, capturing ideas and omitting a part of the script without deleting it. See [Synopses], page 4, [Notes], page 4, and [Comments], page 5. These elements are not usually included in the formatted output.

All of Fountain Mode's capabilities are accessible via the menu, and all customization is performed via the `Fountain` option group. See ⟨undefined⟩ [(emacs) Customization Groups], page ⟨undefined⟩.

Script Elements

The central idea of writing a script in Fountain is that you should be able to just start writing — as long as your text looks like a script, you should get correctly formatted output.

> **n.b.:** Fountain Modeuses the Fountain 1.1 syntax specification. While some programs use custom additions to the specification, for the greatest cross-compatibility, stick with the 1.1 spec.

Metadata

A Fountain script usually starts with some metadata stating the basic information about the script. These are colon-separated key-value pairs at the start of a file:

```
key: value
```

Other Fountain-compatible programs may refer to this as "title page" info, but metadata can store information not related to or present on the title page.

Calling `auto-insert` will guide you through adding the most common Fountain metadata, prompting with likely defaults:

- `title` is pretty obvious, and will default to base-name of the current buffer.
- `credit` is not actually your name, but the type of credit given to the `author`, e.g. `written by` or `screenplay by`.
- `author` is you or your writing team, and defaults to the value of variable `user-full-name`.
- `format` will override the value of `fountain-default-script-format` for the current script. Accepted values are: `screenplay`, `teleplay`, `stageplay`.
- `date` defaults to the current date in your locale's "preferred" format.
- `source`, for if you're writing an adaptation.
- `contact` is useful if the writer wants readers to ever get in touch to make the film/play!

All metadata keys can take either a single line of input following the colon, or a newline followed multiple lines indented with whitespace:

```
author:
Mary Maryland
and
Alan Smithee
```

Metadata keys are case-insensitive, e.g. '`Title`' is the same as '`title`'.

Section Headings

Section headings are lines beginning with `#` (number sign). There are five levels of section headings, with additional `#` characters demoting the outline level of a heading.

```
# Top-Level Heading

## Sub-Heading

### Sub-Sub-Heading
```

See [Outlining], page 6.

Scene Headings

A scene begins with a scene heading.

Scene headings begin with a prefix, specified in `fountain-scene-heading-prefix-list`, which defaults to:

```
INT, EXT, EST, INT./EXT., INT/EXT, I/E
```

Each prefix may be followed by a dot and/or a space, so the following are equivalent:

```
INT HOUSE - DAY

INT. HOUSE - DAY
```

`fountain-scene-heading-prefix-list` [User Option]
: This options allows setting your own scene heading prefixes. These entries are case-insensitive.

`fountain-scene-heading-suffix-separator` [User Option]
: This option allows setting the separator (a regular expression) between the scene location and time of day. The default ‘ `--?` ’ allows either one or two dashes.

See [Scene Heading Completion], page 13.

Action

Action is the easiest Fountain element — anything that isn't parsed as another element is considered action.

Sometimes you may write some action that will be unintentionally parsed as dialogue, for example:

```
Jack examines his shopping list...

BLACK SHIRT
BLACK PANTS
EXPLOSIVES
MAP
```

Here `BLACK SHIRT` would be parsed as a character, who then shouts “BLACK PANTS!...” We don't want that. To prevent this, prefix the first line with `!` (exclamation mark). This is known as a “forced” element.

```
!BLACK SHIRT
BLACK PANTS
EXPLOSIVES
MAP
```

Characters

To write a character name to cue some dialogue, type the name in uppercase (ALL-CAPS).

```
JOLENE
```

The next line will be parsed as dialogue.

Sometimes you might want to include lowercase letters in a character name. To force a line as a character name, prefix it with `@` (at sign).

```
@JOLENE McCLOGGS
```

Some Fountain tools may not parse a line as a character name if the extension contains lowercase letters, requiring the `@` prefix.

```
@JOLENE (cont'd)
```

If you are just writing the character's name within action, type the character's name normally.

```
Jolene
```

If you prefer to write character names within action in uppercase, that's fine too. The following will still be parsed as action.

```
JOLENE throws the chair at PHILIP.
```

See [Character Name Completion], page 13.

Dialogue

Dialogue is any text following [Characters], page 3. Just enter a newline and the next text entered will be parsed as dialogue.

```
JOLENE
Have you seen trouble? I seem to have misplaced it.
```

Parentheticals

Anytime the writer types a (parens) inside of a dialogue block, the text auto-formats into the correct position.

```
JOLENE
(concerned)
Have you seen trouble? I seem to have misplaced it.
```

See [Autocompletion], page 13.

Transitions

Sorry, this node is not yet written. You can help by submitting a patch!

Notes

Sorry, this node is not yet written. You can help by submitting a patch!

Synopses

A synopsis is a handy way to detail what a scene or section is about. A synopsis element is simply a line beginning with `=` (equals sign).

```
INT. FISHING TRAWLER - DAY
= The men eat the shark they caught.
```

Synopses are not included by most export tools.

Center Text

Sorry, this node is not yet written. You can help by submitting a patch!

Comments

Sorry, this node is not yet written. You can help by submitting a patch!

Outlining

There are five levels of section headings. Scene headings count as the sixth level headings.

Cycle an individual subtree visibility with *TAB*. Cycle global outline visibility with *S-TAB* (shift-tab) or *C-u TAB*.

Acts, Sequences, Sections, and Scenes can be given meaningful titles, giving the writer a bird's eye view of the story and structure at a moments notice.

There are many ways to approach writing a screenplay. Here's one example of how to use Fountain Modeto Outline a script.

```
# Act

= Synopsis of an Act. A short summary of all the crazy things that
happen for the next 30-60 pages.

[[ Act One Note. Useful for character motivation and obstacles. Ideas to
remember, etc.]]

## Sequence

= Synopsis of a Sequence.

[[ Sequence Notes. A sequence can be thought of as a series of several
scenes that make up their own mini-story. ]]

INT. SCENE - NIGHT

= Synopsis of a scene.

[[ Notes to remember for a scene, such as the following:
- Who wants what from whom?
- What are they willing to do get what they want?
- What happens if they don't get it? ]]
```

Experimenting with different structures can be accomplished by folding a Section Heading and moving the section to the new desired location using keyboard shortcuts.

Note: all text contained within the fold is carried to the new location.

Just as there are many ways to tell a story, there are many ways to outline a script. Because Fountain Modeuses plaintext, it does not force the writer into a single way of working. Possible structures are limited only by one's imagination.

`fountain-insert-section-heading` [Command]

Bound to *M-RET*, insert an empty section heading at the current outline level.

`fountain-outline-to-indirect-buffer` [Command]

If you want to focus on discrete sections of your script you can open these in indirect buffers. Bound to *C-c C-x b*, this command clones the current section or scene to indirect buffer.

See ⟨undefined⟩ [(emacs) Indirect Buffers], page ⟨undefined⟩,

`fountain-pop-up-indirect-windows` [User Option]

Set this to control how indirect buffer windows are opened. Sometimes you might want to limit your focus to one sequence, other times you might want to look at two scenes in windows side-by-side. Set this option to spawn a new window.

Navigation

`fountain-outline-next` [Command]
: Bound to `C-M-n`, move to the next visible heading line. Also `fountain-outline-previous`, bound to `C-M-p`.

`fountain-outline-forward` [Command]
: Bound to `C-M-f`, move forward to the ARG'th subheading at same level as this one. Also `fountain-outline-backward`, bound to `C-M-b`.

`fountain-outline-beginning` [Command]
: Bound to `C-M-a`, move to the beginning of the current subtree.

`fountain-outline-up` [Command]
: Bound to `C-M-u`, move to the visible heading line of which the present line is a subheading.

`fountain-forward-character` [Command]
: Bound to `M-n`, goto Nth next character (or Nth previous is N is negative). Also `fountain-backward-character`, bound to `M-p`.

`fountain-forward-page` [Command]
: Bound to `C-x ]`, move to Nth next page (or Nth previous if N is negative). Also `fountain-backward-page`, bound to `C-x [`.

`fountain-goto-scene` [Command]
: Bound to `M-g s`, move point to Nth scene in current buffer.

`fountain-goto-page` [Command]
: Bound to `M-g p`, move point to Nth page in current buffer.

Syntax Highlighting

Sorry, this node is not yet written. You can help by submitting a patch!

Element Aligning

Sorry, this node is not yet written. You can help by submitting a patch!

Text Emphasis

Text can be underlined, italic, bold, or a combination thereof.

Underlined text is surrounded by `_underscores_`.

Italic text is surrounded by `*single asterisks*`

Bold text is surrounded by `**double asterisks**`

For the writer pursists who want to work the way our ancestors did on typewriters, stick to underlining.

Do What I Mean

Like many screenwriting programs, in Fountain Modepressing *TAB* will do the most convenient thing based on context.

The most common use is triggering autocompletion. If the point is at a blank line or the end of a non-blank line, pressing *TAB* will call `completion-at-point`. See [Autocompletion], page 13.

In Fountain Mode, *TAB* is also used to control outline visibility. So if point is at a scene or section heading, it will cycle visibility of that scene of section between collapsed and expanded. To allow for more control over outline cycling, if *TAB* is prefixed with `ARG`, call `fountain-outline-cycle` and pass `ARG`. See [Outlining], page 6.

TAB also helps working with parentheticals. If the point is at a blank line within dialogue, it will insert a parenthetical; if the point is inside an empty parenthetical, it will delete it, or if inside a non-empty parenthetical, move to a newline.

If the point is at or inside a note, *TAB* will cycle visibility of that note between collapsed and expanded.

This all might seem complicated, but the idea is by covering all the cases you don't have to think about it.

`fountain-dwim` [Command]

> This is the command you'll use the most. Bound to *TAB*, it will perform the most convenient action based on the current context.

Autocompletion

One of the nice things about using a dedicated screenwriting program is that it helps you type less of the things you need to type a lot. Fountain Modeprovides autocompletion for scene headings and character names.

`fountain-completion-update` [Command]
: This command, bound to *C-c C-x* `a` will update the completion candidates for current buffer.

Scene Heading Completion

If the line has a partial scene heading, i.e. it begins with a prefix from `fountain-scene-heading-prefix-list` like so:

```
INT. |
```

TAB will offer completions of previously used locations.

If the cursor is at the time-of-day, like so:

```
INT. SUBMARINE - |
```

TAB will offer completions from `fountain-scene-heading-suffix-list`.

Character Name Completion

The most basic use of this is when pressing *TAB* on an empty line. If there's an empty line above, this will offer to autocomplete a character name. Character names are suggested in the order:

1. the second-to-last previously speaking character within the current scene, i.e. a character's conversational partner;
2. the last speaking character within the current scene, i.e. a character continuing speaking;
3. the remaining characters in the script in order of frequency (default if there are not yet speaking characters in the current scene).

TAB will also offer character name completion if a line has a partial character name.

When the cursor is after a character name and opening parenthesis, *TAB* will offer completions from `fountain-character-extension-list` plus `fountain-continued-dialog-string`.

```
MARY (|
```

When the cursor is at an empty line within dialogue, *TAB* will add an empty parenthetical.

```
MARY
|
I'm hungry.
```

Likewise, if the cursor is within an empty parenthetical, *TAB* will remove the parenthetical.

```
MARY
(|)
I'm hungry.
```

When the cursor is at the end of a non-empty parenthetical, either inside or outside the closing parenthesis, *TAB* will move to the beginning of the next line if the next line is non-empty, otherwise it will insert a newline.

```
MARY
(angry|)
I'm hungry.
```

When the cursor is at the end of a non-empty line of dialogue, and the value of `fountain-dwim-insert-next-character` is non-nil, *TAB* will insert an empty line and the second-to-last previously speaking character.

```
MARY
(angry)
I'm hungry.|
```

The cursor will be left at the end of the next character, allowing successive presses of *TAB* to cycling through additional character completion candidates.

```
MARY
(angry)
I'm hungry.

JOHN|
```

Scene Numbering

Sorry, this node is not yet written. You can help by submitting a patch!

Pagination

Sorry, this node is not yet written. You can help by submitting a patch!

`fountain-count-pages` [Command]

Bound to *C-c C-p*, return both the current page and the total page count of the current buffer.

Exporting

Exporting a script in Fountain Modeis handled by one or more external command-line tools. Here are some recommended export tools:

- Afterwriting (`https://github.com/ifrost/afterwriting-labs`)
- Wrap (`https://github.com/Wraparound/wrap`)
- Screenplain (`https://github.com/vilcans/screenplain`)
- Textplay (`https://github.com/olivertaylor/Textplay`)[1]

By defining an "export profile", you can easily interface with an external tool from within Emacs. A profile is essentially a shell command, interpolated with a selection of values:

- `%b` is the `buffer-file-name`
- `%B` is the `buffer-file-name` sans extension
- `%n` is the `user-full-name`
- `%t` is the title (See [Metadata], page 2.)
- `%a` is the author (See [Metadata], page 2.)
- `%F` is the current date in ISO format
- `%x` is the current date in your locale's "preferred" format

`fountain-export-command` [Command]
: This command, bound to `C-c C-e`, will prompt for an export profile. The first profile is considered default, so just hitting RET is a quick way to export in your usual way.

`fountain-export-view` [Command]
: This command, bound to `C-c C-v`, attempts to open the last exported file. This works by finding the other most recently modified file in the current directory matching the current file base-name.

[1] Requires PrinceXML (`https://www.princexml.com`) for PDF export.

Bugs and Feature Requests

To report bugs and feature requests that affect the Debian package, please use the following method:

```
$ sudo apt install reportbug
$ reportbug elpa-fountain-mode
```

Contact

If you run into any trouble using Fountain Mode, or you have a feature request, you can email the maintainer Paul W. Rankindirectly at `pwr@bydasein.com`.

For bugs, please ensure you can reproduce with:

```
$ emacs -Q -l fountain-mode.el
```

You can also try the `#emacs` IRC channel on Libera.chat (`https://libera.chat`) where Fountain Mode's maintainer uses the nickname `rnkn`, but please note that most other users on the channel are unlikely to be screenwriters.

Known issues are tracked with `FIXME` comments in the source.

Financial reward is not a consideration in maintaining Fountain Mode, but if you would still like to support development, donations are graciously accepted via GitHub (`https://github.com/sponsors/rnkn`) or Liberapay (`https://liberapay.com/rnkn/`).

Indexes

Key Index

C

M

S

T

Index of Commands

Index of User Options