Emacs, org-mode, and GTD

Richard Lewis

Goldsmiths College, University of London

14 July 2009

Outline

1. Emacs
2. org-mode
3. Getting Things Done
4. GTD with org-mode

Goldsmiths
UNIVERSITY OF LONDON

What is Emacs?

- Emacs (for "Editing MACroS") is a text editor
- Developed by Richard Stallman at MIT
- Available on numerous platforms, including GNU, Mac OS, and Windows
- GNU Emacs is one of the most common varieties
- Emacs is powerful
- But with great power comes great difficulty
- Emacs is about return on investment; invest a lot into it, and you'll get a lot out of it

Goldsmiths
UNIVERSITY OF LONDON

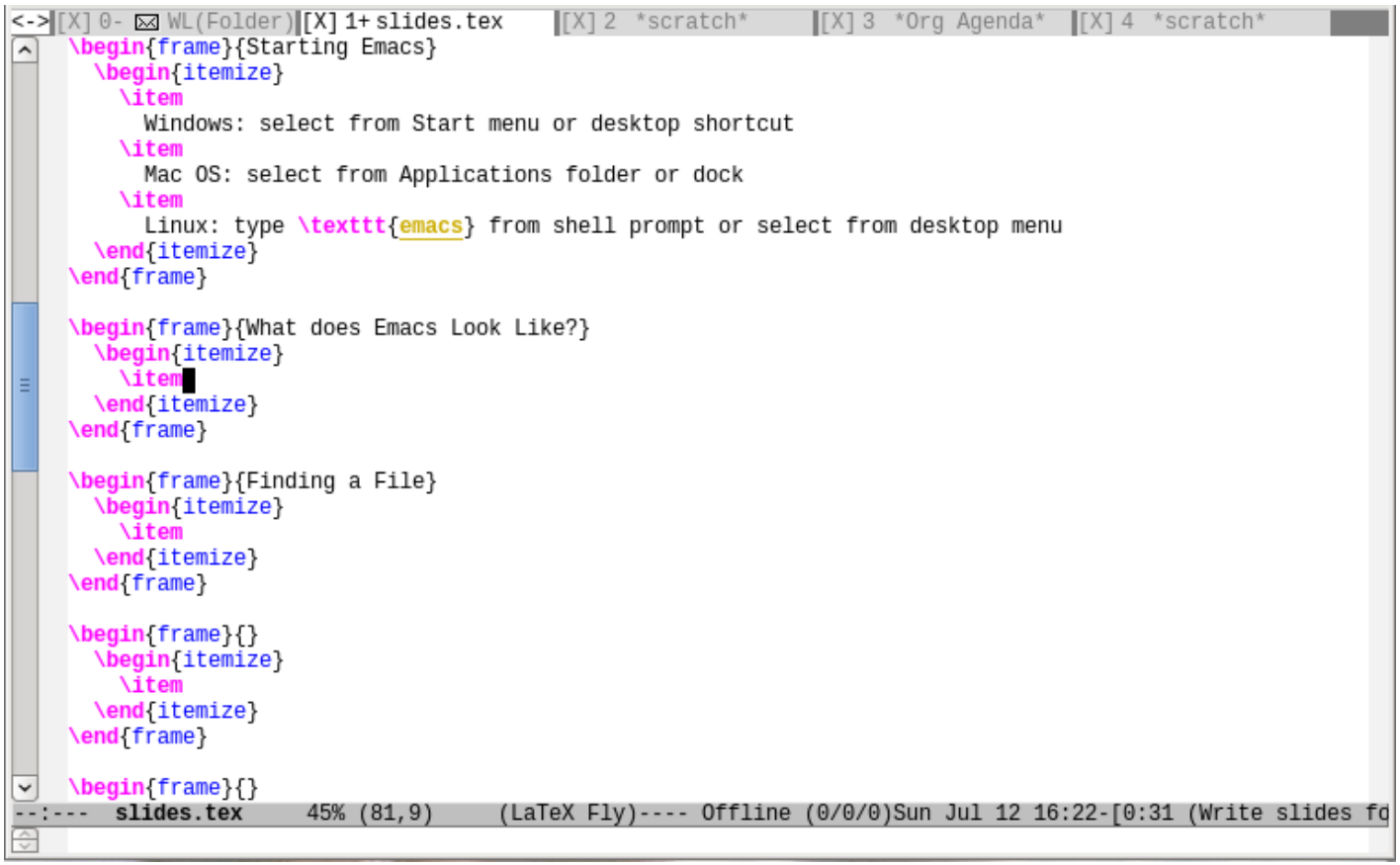
Getting Emacs

- Windows: see `http://www.gnu.org/software/emacs/windows/ntemacs.html`
- Mac OS: see `http://aquamacs.org/`
- Linux: distribution dependent
- Emacs is available on College desktops

Starting Emacs

- Windows: select from Start menu or desktop shortcut
- Mac OS: select from Applications folder or dock
- Linux: type `emacs` from shell prompt or select from desktop menu
- Emacs starts with a splash screen; information on how to get help
- For now, dismiss this by pressing `C-x k`

What does Emacs Look Like?



Basic Emacs view

- You are now in the `*scratch*` buffer, use as a jotter

Goldsmiths
UNIVERSITY OF LONDON

Commanding Emacs I

- Emacs includes numerous text editing commands
- Many are associated with *keybindings*
- Keybinding descriptions: `C-g` means hold down Ctrl, press `g`, release Ctrl
- `C-x C-s` means hold down Ctrl, press `x`, press `s`, release Ctrl
- `C-x f` means hold down Ctrl, press `x`, release Ctrl, press `f`
- `M-q` means hold down Meta (usually Alt), press `q`, release Meta
- Watch out for `M->` (for example); you'll need Shift too
- Meta can also be invoked with `ESC`. In this case, press and release `ESC`, press other key

Commanding Emacs II

- Keybindings are about *finger habits*; they take a few minutes to learn, but they're very easily moved to muscle (rather than conscious) memory
- `M-x` allows you to execute any interactive command by name
- Oops! Press `C-g` to get out of anything you didn't mean (e.g. minibuffer commands you called accidentally)
- You'll need to get out of the habit of reaching for Escape, and learn to use `C-g` instead

Goldsmiths
UNIVERSITY OF LONDON

Keybinding Conventions

- `C-x ...` is often something to do with controlling Emacs itself (buffers, windows)
- `C-c ...` is often something to do with a major mode (i.e. not core Emacs)
- `C-h ...` is always help
- `M-...` is usually like a `C-...` binding, but bigger or more unusual (or sometimes opposite)
- `C-u [some other keybinding]` usually means, "do the other keybinding but ask for the required argument"
- `C-u` can also accept a numerical argument and have the following keybinding repeat
- `C-c C-c` is usually send, post, store

Goldsmiths
UNIVERSITY OF LONDON

Basic Editing

- In many *modes* (including *Fundamental*) alphanumeric and punctuation keys simply insert characters
- Moving the cursor: arrow keys; `C-f`, `C-b`, `C-p` (previous line, or up), `C-n` (next line, or down)
- For completeness, `M-x forward-char RET`, `M-x backward-char RET`, `M-x previous-line RET`, `M-x next-line RET`
- `M-f` (`forward-word`), `M-b` (`backward-word`)
- Delete or `C-d`, `delete-char`
- Backspace, `delete-backward-char`

Goldsmiths
UNIVERSITY OF LONDON

Buffer Navigation

- Some more useful cursor movement commands:
- `C-a` (or `HOME`) (`move-beginning-of-line`)
- `C-e` (or `END`) (`move-end-of-line`)
- `C-v` (or `PGDN`) (`scroll-up`)
- `M-v` (or `PGUP`) (`scroll-down`)
- `M-<` (`beginning-of-buffer`)
- `M->` (`end-of-buffer`)

Goldsmiths
UNIVERSITY OF LONDON

Finding a File

- `C-x C-f` (`find-file`)
- Command prompts for file path in the *minibuffer*
- Tab completion of paths is available
- You may also visit a non-existent "file" (Emacs will create it for when you attempt to save it)
- `RET` to accept choice

Buffers

- Emacs works with *buffers* of text, stored in memory, displayed on screen
- Buffers may show the contents of a file, and their contents may be written to a file
- For most editing purposes buffers and files feel homogeneous (Emacs remembers the association of a buffer with a file)
- To write the content of a buffer to a file (i.e. save): `C-x C-s`
- To write the content of a buffer to a *different* file: `C-x C-w`
- To close a buffer press `C-x k`

Modes

- Each buffer is in exactly one *major mode*
- Intended for different types of editing tasks, allows buffer-specific keybindings, syntax highlighting, etc.
- Each buffer may also be in zero or more *minor modes*
- Intended to provide editing facilities which may be compatible with major modes or other minor modes (e.g. spell checking, parentheses matching)

Goldsmiths
UNIVERSITY OF LONDON

flyspell-mode

- Activate by calling flyspell-mode command
- Provides an interactive spell checker
- Highlights words it doesn't recognise
- Use `M-TAB` to auto-correct word under cursor
- (Also, `M-$` is bound to `ispell-word` which provides interactive correction, but is not part of `flyspell-mode`)

Goldsmiths
UNIVERSITY OF LONDON

Cursors, Points, and Marks

- One *cursor* per Emacs
- One *point* per buffer
- Zero or one *marks* per buffer
- In the current buffer the cursor is in the same place as the point
- When you visit a different buffer, the cursor moves to the position of the point in that buffer
- Set a mark with: `C-SPACE`

Saving, Killing, and Yanking

- The *region* is the area between the mark and the point
- To select a region: set the mark (`C-SPACE`), move the cursor (and thereby the point) to the end of the desired region
- Activate `transient-mark-mode` to have regions highlighted: `M-x transient-mark-mode RET`
- To cancel the current region: `C-g` (as usual)
- To *kill* some text (like *cut*), select a region to kill, press `C-w` (`kill-region`)
- To *yank* some text (like *paste*), press `C-y` (`yank`)
- To *save* some text (like *copy*), select a region to save, press `M-w` (`kill-ring-save`)

Goldsmiths
UNIVERSITY OF LONDON

The Kill Ring

- Emacs puts killed and saved text into the *kill ring*
- The kill ring is more useful than most OS clipboards
- This holding area for killed text retains the last `kill-ring-max` previously killed entries
- Immediately after yanking, you may cycle through the kill ring's contents (using `M-y`) and choose a different kill to yank

Goldsmiths
UNIVERSITY OF LONDON

Incremental Search

- Press `C-s` (`isearch-forward`), start typing your search terms and Emacs begins searching immediately
- Press `RET` to stop searching and leave the cursor where you've got to
- Press `C-g` to give up and go back to where you started
- Press `C-s` (again) to exit interactive search and search for the next occurrence of what you've found so far (repeat)
- `C-r` (`isearch-backward`)

Replacing Text

- Press `M-%` (`query-replace`) to begin replacing text
- Minibuffer prompts for text to replace, and then text to replace it with
- `query-replace` prompts at each occurrence, press `y` to replace, `n` to skip
- In `query-replace` other options include: `!` replace all remaining occurrences without prompting, `q` quit `query-replace`

Goldsmiths
UNIVERSITY OF LONDON

Multiple Buffers

- Emacs allows multiple buffers
- To switch to another buffer: Press `C-x b`, minibuffer prompts for name of buffer (tab completion available)
- After pressing `C-x b`, you can press `RET` immediately to go to the previous buffer you were looking at
- (The `iswitchb` mode provides much better buffer switching, including partial buffer name matching)
- Press `C-x C-b` to see a list of open buffers

Windows

- Emacs also allows you to view more than one buffer at once (or have more than one view on the same buffer)
- By using multiple *windows*
- Press `C-x 2` (`split-window-vertically`) to split the current window vertically
- Press `C-x 3` (`split-window-horizontally`) to split the current window horizontally
- Press `C-x 1` (`delete-other-windows`) to make the current window the only window
- Press `C-x 0` (`delete-window`) to close the current window
- Press `C-x o` (`other-window`) to jump to the next window

Goldsmiths
UNIVERSITY OF LONDON

Window Example

- So with the buffer list just now, you can summon the buffer list (`C-x C-b`)
- jump to its window (`C-x o`)
- select a buffer with the arrow keys
- view it the other window (`o`)
- and make it the only window (`C-x 1`)

Keyboard Macros

- Emacs can record keyboard actions and replay them
- Press `C-x (` (`kmacro-start-macro`) to begin recording macro
- Press `C-x )` (`kmacro-end-macro`) to stop recording macro
- Press `C-x e` (`kmacro-end-and-call-macro`) to execute last recorded macro
- Press `e` to repeat previously executed macro

Goldsmiths
UNIVERSITY OF LONDON

Other Text Facilities

- Paragraph formatting; `M-q` to wrap a paragraph
- Rectangular regions

Getting Help

- To start the Emacs tutorial: `C-h t`
- To view the Emacs splash screen: `C-c C-a`
- To see a list of keybindings available for your current modes: `C-h b`
- To find out what a key combination is bound to: `C-h k [the keybinding]`
- To find the keybinding for a command: `C-h w`

Goldsmiths
UNIVERSITY OF LONDON

Emacs is Extensible

- The core of Emacs is a Lisp interpreter
- All but the most fundamental commands are implemented in Lisp
- Commands may be added, removed, or altered in real time

What else is *scratch* for?

- The *scratch* buffer is not just (or possibly, not really) a jotter
- It's in `lisp-interaction` mode by default, you can execute elisp commands in it
- For example, `(* 524 1287)`
- Place cursor at end of s-expression and press `C-j` to evaluate it
- Other examples: `(transient-mark-mode 1)`, `(delete-selection-mode 1)`
- Even `(defun dot-is-and-cross-ts () ...)`

Goldsmiths
UNIVERSITY OF LONDON

~/.emacs

- Your ~/.emacs may contain commands to run when Emacs starts
- e.g. add-on packages to load: `(require 'foo-bar)`
- e.g. setting variables: `(setq some-var "some value")`
- e.g. `(transient-mark-mode 1)`

What is org-mode?

- org-mode is a major mode for Emacs which enables structured note taking
- Also includes several other modes and utilities for manipulating your notes
- Your notes are stored in plain text which you can keep forever, whatever platform or applications you use in future
- Allows semantic tagging of notes
- Documentation: `M-x org-info RET`

Goldsmiths
UNIVERSITY OF LONDON

Enabling org-mode

- org-mode is distributed as part of Emacs since 22.2
- In *scratch* evaluate `(require 'org)`
- (You can also add `(require 'org)` to your .emacs file)

A buffer in org-mode

- .org files should be associated with org-mode
- In *scratch* evaluate `(add-to-list 'auto-mode-alist '("\\.org\\'" . org-mode))`
- Find a file with a .org extension

Taking Notes

- Begin a line with an asterisk (*) to make a note
- `M-RETURN` to start a new note on the next line
- Notes can be hierarchical
- `M-RIGHT` to increase/lower structural level ("demote" note)
- `M-LEFT` to decrease/raise structural level ("promote" note)

Trees

- The structure of your notes is a tree, each note has either zero or one parents
- Trees may be *folded* (hide the child notes) and *unfolded* (show the child notes)
- With the cursor over a parent note, press `TAB` to unfold its children
- Press `TAB` again to unfold all its descendants
- Press `TAB` a third time to fold them all up again

Links

- org-mode is able to understand various kinds of links
- Web: `http://orgmode.org/`
- File: `file:~/research/biblio.bib`, or `C-u C-c C-l`, tab completion available in both forms
- Email: `mailto:richard.lewis@gold.ac.uk`
- You may also render links with descriptive text: `[[http://www.gold.ac.uk/][Goldsmiths]]`
- Links may be typed manually, or use `C-c C-l`
- To visit a link, with the cursor over the link, `C-c C-o` (behaviour is quite platform dependent)

Goldsmiths
UNIVERSITY OF LONDON

Storing Links

- Function `org-store-link` stores a link to whatever the cursor is on (maybe whole file, part of a file)
- Many Emacs modes (e.g. Wanderlust mail reader, w3m-el web browser) have useful ways of interpreting `org-store-link`
- Many `org-mode` users associate a global keybinding with this function:

```
(define-key global-map "\C-cl" 'org-store-link)
```

(in `*scratch*`)

- Now, pressing `C-c l` in some buffer will store a link
- `C-c C-l` in an `org-mode` buffer inserts a link, use `TAB` to retrieve stored links

Goldsmiths
UNIVERSITY OF LONDON

Export

- org-mode provides functions to export your notes files to various formats including HTML, PDF, and plain text
- `C-c C-e` (org-export) summons export options
- Choose `a` for plain text, `h` for HTML, `b` for HTML in browser, `p` for PDF, etc.

Goldsmiths
UNIVERSITY OF LONDON

What is GTD?

- Getting Things Done is an approach to managing knowledge work proposed by the management consultant David Allen
- Basic premise is the distinction between knowledge work and some other types of work
- In knowledge work, the tasks and their outcomes are not always visible or obvious (cf. mowing the lawn, or repairing a wall)
- So the knowledge worker's main tool (the conscious mind) is continuously busy with irrelevant (though not necessarily unimportant) and distracting jobs
- Whether or not we are engaged in knowledge work, our minds are always busy in this way

Goldsmiths
UNIVERSITY OF LONDON

The Need for GTD

- Allen argues that many knowledge workers are overwhelmed by the things vying for their attention and time
- He argues that they suffer from trying to rely on their memory as storage for pending tasks, projects, ideas, etc.
- When an idea is stored in your head, it keeps distracting you while you try to work on something else
- Allen argues that this leads to stress, procrastination, and disorganisation

The Basics of GTD

- Don't rely on your memory for storing pending tasks
- Instead rely on written lists
- It's absolutely vital that whatever system you implement for your lists you learn to **trust completely**
- This way, your pending tasks will no longer disturb you while you're working on something else
- You'll be confident that everything else you need to pay attention to later is safely recorded

Five Stage Workflow

- Allen's method is a five-stage *workflow*:
- **Collect**: all the incoming ideas, messages, information ("stuff")
- **Process**: all the collected "stuff", deciding what to do about/with it
- **Organise**: all the processed "stuff" into manageable categories
- **Review**: all the organised "stuff", to keep the system functional
- **Do**: the "stuff" that's doable and current / contextual / appealing / etc.

Goldsmiths
UNIVERSITY OF LONDON

Collecting I

- What to collect?
- Allen describes the "incompletes"
- "[y]ou need to collect and gather together placeholders for or representations of all the things you consider incomplete in your world" (Allen 2001, 26)
- It's anything that you think should be different to how it is
- Anything that can be expressed "should", "need to", "ought to", "going to"
- He also describes these things as "open loops"; you can feel them nagging at you

Collecting II

- They all need to be in one place for later processing
- You need to have a container for them
- He says these collection containers are successful when:
  - They contain everything that was in your head
  - You have only a few of them
  - You empty them regularly

Processing I

- "What do you need to ask yourself (or answer) about each e-mail, voice mail, memo, or self-generated idea that comes your way?" (Allen, 31)
- What is it?
- Is it actionable? Does it represent something which requires a definite action?

Processing II

- If it's not actionable, either:
  - It's rubbish, so get rid of it
  - No action is required now, but it might require an action in the future, so "incubate" it
  - It contains useful reference information, so file it

Processing III

- If it is actionable:
  - What's the *next action* associated with it?
  - If it's associated with a project, you need to add its outcome to a projects list
  - **Do it**: if it takes less than two minutes
  - **Delegate it**: if someone else should do it, add to *waiting* list
  - **Defer it**: if it'll take longer than two minutes, add it to your *next actions* list or to your *calendar* if it's for a specific date

Organising

- Allen identifies eight required storage areas resulting from the processing phase (seven given here):
- For non-actionable items: *rubbish*, *incubation*, *reference*
- For actionable items: *projects*, *next actions*, *calendar*, *delegated* or *waiting* list
- Items in your *next actions* list should also have a *context*, describing what class of action they are, or where you can do them
- e.g. Phone call, E-mail, Computer, Office, Home, Buy, Library

Projects

- Allen defines a project as any outcome which requires more than one step
- He describes the requirement for a "stake in the ground" which groups a list of related *next actions*
- He argues that, if this placeholder is not present, the project will become something you have to store in your memory and will begin to demand your attention
- "You don't actually *do* a project; you can only do action steps *related* to it." (Allen, 38)

Review

- Allen argues that you should review your actionable lists often to keep them both fresh and off your mind
- Order of frequency of review: *calendar*, *next actions*, *waiting*, *projects*
- You will likely review your *next actions* between every period of activity

Weekly Review

- Allen's key to success of his method is the *Weekly Review*
- Here you review all your actionable lists thoroughly
- But also your *incubation* (or *someday/maybe*) list
- He summarises:
  - Gather and process all your "stuff" (including bits of paper, books, broken gadgets)
  - Review your system
  - Update your lists
  - Get clean, clear, current, and complete

Do

- Allen argues that his system is about giving the user the best information to decide what to do at any point during the day
- He concedes that actually deciding is, "an intuitive call"
- He describes four criteria for deciding which *next action* to do at any given time:
  - Context (what type of action it is, or where you need to be)
  - Time available
  - Energy available
  - Priority
- He describes unplanned-for activities (phone calls received, visitors) as being actions which, by default, usually acquire a higher priority than whatever else you'd planned to do with the time

Goldsmiths
UNIVERSITY OF LONDON

So how can org-mode help with GTD?

- Its facility to tag notes
- Its modes for generating various views of your note data
- Its integration with remember-mode to *collect* incoming "stuff" easily and discretely

Setting Up I

- Create a directory to store your org files in (e.g. "~/org")
- Find a file called "notes.org" in that directory
- This will be your main dumping ground for "stuff"
- Add the following configuration text, then close and reload the notes.org for the configuration to take effect:

```
* conf
#+SEQ_TODO: TODO(t) STARTED(s) WAITING(w) APPT(a) |DONE(d) CANCELLED(c) DEFERRED(f)
#+TAGS: HOME(h) OFFICE(o) EMAIL(e) PHONE(p) READ(r)
```

- Evaluate: `(setq org-directory "~/org")`
- Evaluate: `(setq org-agenda-files '("~/org"))`
- Evaluate: `(require 'remember)`
- Evaluate: `(require 'org-remember)`
- Evaluate: `(setq remember-annotation-functions '(org-remember-annotation))`

Goldsmiths
UNIVERSITY OF LONDON

Setting Up II

- Evaluate: `(setq remember-handler-functions '(org-remember-handler))`
- Evaluate: `(add-hook 'remember-mode-hook 'org-remember-apply-template)`
- Evaluate: `(setq org-default-notes-file (concat org-directory "/notes.org"))`
- Evaluate: `(define-key global-map "\C-cr" 'org-remember)`

Goldsmiths
UNIVERSITY OF LONDON

Collecting

- `remember-mode` provides a simple an non-intrusive way of making notes in Emacs while you're working on something else
- We have configured `org-mode` to use `remember-mode` to store incoming ideas
- Press `C-c r` to make a note
- The new note will contain a link to whatever you were looking at (delete this if you like)
- Enter: `* Wireless mouse stopped working`
- Press `C-c C-c` to store it

Goldsmiths
UNIVERSITY OF LONDON

Processing

- org-mode provides several different types of note tagging
- In an org-mode buffer, press `C-c C-t` to select a TODO status
- In notes.org we set up various TODO statuses which roughly follow Allen's guidelines
- TODO is on the *next actions* list; STARTED is ongoing; WAITING is on the *waiting* list; APPT is a timed event.
- Often, processing can be done at the same time as collecting
- You can mark a note as TODO (`C-c C-t`) when you enter it with remember-mode

Goldsmiths
UNIVERSITY OF LONDON

Organising

- Another type of tagging org-mode provides is "tags"
- In our notes.org file we have the following available tags: HOME, OFFICE, EMAIL, PHONE, READ
- To tag a note when viewing it in an org-mode buffer, press `C-c C-c`, select the tag mnemonic, press `RET`
- You can also group notes underneath project headers
- I'm not as strict as Allen with projects. I use separate org files for larger projects, and neglect to group small projects

Goldsmiths
UNIVERSITY OF LONDON

Organising II

- org-mode also understands dates and times
- You can insert a date with `C-c .`
- You can set a deadline for a TODO with `C-c C-d`
- You can schedule a TODO for a day with `C-c C-s`

Reviewing

- org-mode provides the *Agenda mode* to view your TODOs in various ways
- Evaluate: `(global-set-key "\C-ca" 'org-agenda)`
- Press `C-c a`, press `a` to see your TODOs in a weekly planner
- Use left and right arrow keys to move between weeks, and `.` to go to today
- Use `r` to refresh after making changes to any .org files

Goldsmiths
UNIVERSITY OF LONDON

Reviewing II

- Press `C-c a`, press `t` to see your TODOs in a list, grouped by org file
- If you keep one org file per project, this grouping becomes per project
- To filter the TODOs by tag, press `\ TAB`, type the name of the tag (tab completion available), `RET`
- If you use `org-mode`'s tags to store Allen's *context* information, this facility gives you easy access to a list of TODOs relevant to your current context
- Press `\ \` to remove the filter

Goldsmiths
UNIVERSITY OF LONDON

Reviewing II

- When viewing a project .org file, press `C-c a L` to view all your activity on that project on a timeline

Doing

- org-mode allows you to *clock in* to a TODO
- With the cursor over the TODO in its org-mode buffer, press `C-c C-x C-i`
- To *clock out*, press `C-c C-x C-o`
- To cancel the current clock: `C-c C-x C-x`
- You can also clock in to a TODO from the Agenda mode by pressing `I` with the cursor over the TODO, and clock out by pressing `O`

Goldsmiths
UNIVERSITY OF LONDON

Final Notes

- Occasionally, people need to quit Emacs
- If you ever find that this is necessary, use `C-x C-c` (`save-buffers-kill-emacs`)