

Never compile this.

```
;; -*- no-byte-compile: t; -*-
```

1. Org-Mode Fundamentals

ID: orgmode:gcr:vela:7E71A009-4DD3-4296-9851-293BC71D2DFF

Configure Org-Mode core functionality to compile this system.

Sysop is likely to use this periodically.

Start EMACS with this command:

```
emacs --debug-init --no-init-file --no-splash --background-color white --foreground-color bl  
(load-file "~/src/help/.org-mode-ecm.emacs.el")
```

(a) Literate Programming

ID: orgmode:gcr:vela:8510C876-F644-4804-9F87-54A0B44DBA6A

This system enables you to perform 3 Literate Document activities

- Tangling
- Evaluating
- Weaving

Combined they provide a rich Literate Programming environment.

These activities are not performed interactively by the user. They are automatic operations that occur as a result of the configuration by the document itself.

The following is the guide for the default configuration of this system and how it behaves.

Key:

- Columns
 - S** Source document modified?
 - T** Destination tangled-document modified?
 - W** Destination weaved-document modified?
 - C** Source-Block Evaluation occurred?
 - O** Org-Macro Expansion occurred?

Activity	S	T	W	C	O
Tangling	F	T	F	F	F
Evaluating	T	F	F	T	F
Weaving	F	F	T	F	T

They are separate and distinct operations.

“Programming” is logically an activity that is the combination of these 3 activities. It is interactively performed by Sysop. It is not a distinct or isolated operation. Results of one activity exist here and serve as inputs to another activity.

i. Helper Functions

ID: orgmode:gcr:vela:B14776FD-6835-4D1D-BCD3-50D56555423C

Help configure Org-Mode.

```
(defun help/set-org-babel-default-header-args (property value)
  "Easily set system header arguments in org mode.
```

PROPERTY is the system-wide value that you would like to modify.

VALUE is the new value you wish to store.

Attribution: URL http://orgmode.org/manual/System_002dwide-header-arguments.html#Sy

```
(setq org-babel-default-header-args
  (cons (cons property value)
        (assq-delete-all property org-babel-default-header-args))))
```

```
(defun help/set-org-babel-default-inline-header-args (property value)
  "See 'help/set-org-babel-default-header-args'; same but for inline header args."
  (setq org-babel-default-inline-header-args
    (cons (cons property value)
          (assq-delete-all property org-babel-default-inline-header-args))))
```

```
(defun help/set-org-babel-default-header-args:R (property value)
  "See 'help/set-org-babel-default-header-args'; same but for R.
```

This is a copy and paste. Additional languages would warrant a refactor."

```
(setq org-babel-default-header-args:R
  (cons (cons property value)
        (assq-delete-all property org-babel-default-header-args:R))))
```

```
(defun help/set-org-babel-default-header-args:ditaa (property value)
  "See 'help/set-org-babel-default-header-args'; same but for ditaa.
```

This is a copy and paste. Additional languages would warrant a refactor."

```
(setq org-babel-default-header-args:ditaa
  (cons (cons property value)
        (assq-delete-all property org-babel-default-header-args:ditaa))))
```

```
(defun help/set-org-babel-default-header-args:dot (property value)
  "See 'help/set-org-babel-default-header-args'; same but for dot.
```

This is a copy and paste. Additional languages would warrant a refactor."

```
(setq org-babel-default-header-args:dot
  (cons (cons property value)
```

```

      (assq-delete-all property org-babel-default-header-args:dot))))

(defun help/set-org-babel-default-header-args:plantuml (property value)
  "See 'help/set-org-babel-default-header-args'; same but for plantuml.

This is a copy and paste. Additional languages would warrant a refactor."
  (setq org-babel-default-header-args:plantuml
    (cons (cons property value)
      (assq-delete-all property org-babel-default-header-args:plantuml))))

(defun help/org-toggle-macro-markers ()
  (interactive)
  (setq org-hide-macro-markers (not org-hide-macro-markers)))

(defun help/org-prp-hdln ()
  "Visit every Headline. If it doesn't have an ID property then add one and
  assign it a UUID. Attribution: URL
  'http://article.gmane.org/gmane.emacs.orgmode/99738'. It is OK to leave the
  colon separator in here because these are never used as Source-Blocks and
  the rest of the code expects the colon separator."
  (interactive)
  (save-excursion
    (goto-char (point-min))
    (dolist (p (nreverse
      (org-element-map (org-element-parse-buffer 'headline) 'headline
        (lambda (headline) (org-element-property :begin headline))))))
      (goto-char p)
      (org-id-get-create))
    (save-buffer)))

(defun help/org-id-new ()
  "Re-purposing 'org-id' hit a snag when colons were forbidden in Source-Block
  names. Adding support for a user-defined Org-Id separator would have fixed
  this but with no benefit to Org-Id. So this function removes the colon
  instead.
  "
  (interactive)
  (let* ((gend (org-id-new))
    (newid (replace-regexp-in-string ":" "_" gend)))
    newid))

(defun help/org-prp-src-blk ()
  "If it doesn't have a NAME property then add one and
  assign it a UUID. Attribution: URL 'http://article.gmane.org/gmane.emacs.orgmode/
  (interactive)
  (help/org-2every-src-block
    #'(lambda (element)
      (if (not (org-element-property :name element))

```

```

        (let ((i (org-get-indentation)))
          (beginning-of-line)
          (save-excursion (insert "#+NAME: " (help/org-id-new) "\n"))
          (indent-to i)
          (forward-line 2))))))

(defconst help/org-special-pre "~\s*#[+]" )

(defun help/org-2every-src-block (fn)
  "Visit every Source-Block and evaluate 'FN'."
  (interactive)
  (save-excursion
    (goto-char (point-min))
    (let ((case-fold-search t))
      (while (re-search-forward (concat help/org-special-pre "BEGIN_SRC") nil t)
        (let ((element (org-element-at-point)))
          (when (eq (org-element-type element) 'src-block)
            (funcall fn element))))))
    (save-buffer)))

(defun help/org-babel-demarcate-block ()
  "Add a NAME property then assign it a UUID."
  (interactive)
  (org-babel-demarcate-block)
  (insert "#+NAME: " (help/org-id-new))
  (beginning-of-line)
  (insert "\n"))

```

ii. Identity

ID: orgmode:gcr:vela:25F4226F-2EB2-48EC-A4D5-56DD5CCC753E

A Headline's primary key is ID. Use org-id to manage it.

(require 'org-id)

In Links: Never use ID or CUSTOM_ID; always use the file name and text to make it accessible outside of Emacs.

(setq org-id-link-to-org-use-id 'nil)

Make sure that ID is always unique, portable, and easy to maintain by

- Using an acceptable prefix
 - Memorable
 - * So you can remember where you created it and when
 - * So you can share it and let the recipient know (in theory useful)
 - * So you can enable a non Emacs/Org-Mode user to work with the tangled code referencing it's origin

– Valid

- * Must be both L^AT_EX label and XHTML identifier compliant

- org-lint checks for this

- Include the current login
- Include the current domain
- Use a UUID

```
(setq org-id-prefix (concat "org_" (user-real-login-name) "_" (format-time-string "%Y-%m-%d_%H-%M-%S")))
(setq org-id-method 'uuid)
```

iii. Tangling

```
ID: orgmode:gcr:vela:267EEDED-1367-405F-807C-B3C489045704
```

ID and NAME are essential for successful LP using org-babel-tangle-jump-to-org.

```
(add-hook 'org-babel-pre-tangle-hook #'help/org-prp-hdln)
(add-hook 'org-babel-pre-tangle-hook #'help/org-prp-src-blk)
```

There is a way to disable property inheritance that speeds up tangling a lot. This is only for user-defined properties; **not** Org-Mode properties.

The problem is that you lose property inheritance which is unacceptable. Never, never allow that. Its inconsistent with how Org-Mode works.

```
(setq org-babel-use-quick-and-dirty-noweb-expansion nil)
```

Assume that tangled document always live within the same directory structure as their origin document.

```
(setq org-babel-tangle-use-relative-file-links t)
```

- Post tangle actions

– Indentation

- * At first glance this is surprising! The author should be responsible for the indentation, right? Yes, that is right. But there is one exception: using :noweb-ref source block concatenation. It is powerful and elegant. But the source blocks are indented on their own line. It forces any reader format it to make any sense of it. That is a poor start to using the tangled files. So tangled files get indented.

```
(defun help/org-babel-post-tangle-hook-fn ()
  (interactive)
  (indent-region (point-min) (point-max) nil)
  (save-buffer))
(add-hook 'org-babel-post-tangle-hook #'help/org-babel-post-tangle-hook-fn)
```

A. comments

```
ID: orgmode:gcr:vela:49787FC5-CAA7-466B-B742-0F38973E070B
```

Toggle insertion of comments in tangled code files

Provide as much information as possible in the tangled artifact about the origin artifact.

(help/set-org-babel-default-header-args :comments "noweb")

B. mkdirp

ID: orgmode:gcr:vela:B0F9A321-3B69-46BB-B512-0AF3C663A4C0

Toggle creation of parent directories of target files during tangling

(help/set-org-babel-default-header-args :mkdirp "yes")

C. no-expand

ID: orgmode:gcr:vela:90170E6A-AA1A-44EA-9BF8-1A6AA38FD224

Turn off variable assignment and noweb expansion during tangling

Configuration likely per Source-Block or System.

D. noweb

ID: orgmode:gcr:vela:E12B48AB-68E8-4515-89E3-30A16FB6FD22

Toggle expansion of noweb references

Expand noweb references in source-blocks before:

Activity	Expand
Tangling	T
Evaluating	T
Weaving	F

This embraces the notion that you are telling the right thing to the computer and the right thing to the human. By the time you get to exporting, you ought to refer to the generated document.

(help/set-org-babel-default-header-args :noweb "no-export")

E. noweb-ref

ID: orgmode:gcr:vela:2836D0AA-5DBA-48AC-A338-B47002DE8D7F

Specify block's noweb reference resolution target

Configuration likely per Source-Block or System.

F. noweb-sep

ID: orgmode:gcr:vela:B1A57D15-6BBF-4E78-A0D9-0B02C283C6B0

String used to separate noweb references

Configuration likely per Source-Block or System.

G. padline

ID: orgmode:gcr:vela:DDE727A6-DDF7-4B61-9063-549614B135F0

Control insertion of padding lines in tangled code files

- `org-babel-tangle-jump-to-org` requires padded lines. This configuration could arguably appear in the “Programming” heading because it impacts operation. It lives here because it **must** occur as part of the Tangling activity so that it can be used in the Programming activity.
- Often I go back and forth on this one. Sometimes it is nicer to have less spaces in generated code when guests are viewing it. When no one else is reading it I love the spaces. Defaulting to what I like.

`(help/set-org-babel-default-header-args :padline "yes")`

H. session

ID: `orgmode:gcr:vela:8219A42A-E90F-418A-8EF0-EB150CF6D730`

Preserve the state of code evaluation

Configuration likely per Source-Block or System.

For some situations, this may be the same for every source block for a particular language. R is a good example.

I. shebang

ID: `orgmode:gcr:vela:542185DD-4FD6-459A-B422-DA7B546FB292`

Make tangled files executable

Configuration likely per Source-Block or System.

J. tangle

ID: `orgmode:gcr:vela:EA716FC9-4A90-4F3E-ABD0-31FEA575C969`

Toggle tangling and specify file name

`(help/set-org-babel-default-header-args :tangle "no")`

K. tangle-mode

ID: `orgmode:gcr:vela:5F0B7157-2DC8-4AFD-8F26-4B21025A5ECE`

Set permission of tangled files

Configuration likely per Source-Block or System.

iv. Evaluating

ID: `orgmode:gcr:vela:ED23FF0B-1F90-435C-9B56-ACA06C1ACAE0`

Org-Mode may use all of the listed languages.

```
(org-babel-do-load-languages
'org-babel-load-languages
'((emacs-lisp . t)
  (org . t)
  ;;
  (C . t)
  (R . t))
```

```

(python . t)
(sass . t)
(scheme . t)
(sql . t)
(js . t)
;;
(latex . t)
;;
(makefile . t)
(shell . t)
;;
(ditaa . t)
(dot . t)
(plantuml . t)))

```

Out of the box Emacs supports js with js-mode. Define language javascript to use js2-mode.

```
(add-to-list 'org-src-lang-modes '("javascript" . js2))
```

A. cache

ID: orgmode:gcr:vela:49B8BFE9-643B-450F-A8A1-20CE3079E215

Avoid re-evaluating unchanged code blocks

Configuration likely per Source-Block or System.

Default no is correct for nearly every scenario.

B. colnames

ID: orgmode:gcr:vela:4D683007-14AE-4A7D-A506-E2301FD32E82

Handle column names in tables

Configuration likely per Source-Block or System.

C. dir

ID: orgmode:gcr:vela:CD1494F1-0A2A-44D0-9955-0D0501AF1539

Specify the default (possibly remote) directory for code block execution

Configuration likely per Source-Block or System.

D. epilogue

ID: orgmode:gcr:vela:CA7F5086-9D4B-4847-9449-3231CE027804

Text to append to code block body

See Prologue.

E. eval

ID: orgmode:gcr:vela:0329BACE-2C99-4BB3-A7A5-7C800EF53FAD

Limit evaluation of specific code blocks

Never evaluate source-blocks or in-line-source-blocks **on export**.

```
(help/set-org-babel-default-header-args :eval "never-export")
```

```
(help/set-org-babel-default-inline-header-args :eval "never-export")
```

Never evaluate in-line-source-blocks **on export**.

```
(setq org-export-babel-evaluate nil)
```

F. file

```
ID: orgmode:gcr:vela:80824708-62AF-4337-A517-828DA22D1FCA
```

Specify a path for file output

Configuration likely per Source-Block or System.

G. file-desc

```
ID: orgmode:gcr:vela:6F9A2745-7118-469E-9FDB-4B327C02E5FA
```

Specify a description for file results

Configuration likely per Source-Block or System.

H. file-ext

```
ID: orgmode:gcr:vela:0716A48E-9227-44FD-B1FA-185DF6545E91
```

Specify an extension for file output

Configuration likely per Source-Block or System.

I. hlines

```
ID: orgmode:gcr:vela:721F4E5E-A343-4D7C-A3A3-12A544B3A273
```

Handle horizontal lines in tables

Configuration likely per Source-Block or System.

J. output-dir

```
ID: orgmode:gcr:vela:D0DDFE88-1B41-4A67-A5F4-88B1B35A7513
```

Specify a directory to write file output to

Configuration likely per Source-Block or System.

One example is a System where **all** intermediate results are stored to individual files.

K. post

```
ID: orgmode:gcr:vela:1A4DEC98-C735-4D88-8261-6AD13C495EF2
```

Post processing of code block results

Configuration likely per Source-Block or System.

L. prologue

ID: orgmode:gcr:vela:3D1780E0-2E6D-428C-916D-BFB10E79C76F

Text to prepend to code block body

Configuration likely per Source-Block or System.

For some situations, this may be the same for every source block for a particular language. The user manual described `gnuplot`, which often shows up on the list and the solution is to `reset` the session.

Another example, say that you've got a bunch of R Source-Blocks and you want to be able to rearrange them as you please. You want to be sure that there are no dependencies between them on bindings created in the workspace. Set prologue to `rm(list = ls())`.

Epilgue works hand-in-hand with this.

M. results

ID: orgmode:gcr:vela:2755571E-113B-436E-9EEC-26618A55A27E

Specify the type of results and how they will be collected and handled

Ways to configure `:results:` (`apply '* (-keep 'cdr '((Collection . 2) (Type . 4) (Format . 7) (Handling . 4)))`) 224.

This system stores the results of evaluation in the source document. It believes that the results are critical to the research.

Keep the document as close to being executable as possible; make it very visible when it is not.

- Collection
 - `value`: Functions have a single result. So do Source-Blocks.
- Type
 - `table`:
 - Tables are the best type because
 - * Dimensions make them human-readable in text.
 - * Work with Babel LP.
 - * Appear as lists to programming languages.
 - * Weaves well.
 - * Inline Source-Blocks disallow tables so use scalars instead.
- Format
 - `drawer`: Enable results replacement
- Handling
 - `replace`: Replace theme each time you evaluate the block.

```
(defconst help/org-sb-results-cfg "value table drawer replace")
(help/set-org-babel-default-header-args :results help/org-sb-results-cfg)
```

Their format will show that they are results. Inline source blocks automatically get formatted as verbatim. For some reason, this only needs to be configured as `replace` to work unlike normal source blocks. Copying the configuration from normal source blocks here breaks the replacement functionality.

```
(defconst help/org-isb-results-cfg "replace")
(help/set-org-babel-default-inline-header-args :results help/org-isb-results-cfg)
```

N. rownames

ID: `orgmode:gcr:vela:B184A507-1B03-4096-A4D8-E50A1DA047DB`

Handle row names in tables

Configuration likely per Source-Block or System.

O. sep

ID: `orgmode:gcr:vela:F1336AAA-68EF-4E87-B253-458103B6FF2F`

Delimiter for writing tabular results outside Org

Configuration likely per Source-Block or System.

P. var

ID: `orgmode:gcr:vela:3B4D638C-82EE-47F3-835C-52B2F03620A0`

Pass arguments to code blocks

- **The** most revealing of the power of Org-Mode's LP offering
- Values-by-reference
 - Table
 - List
 - Source-Block without and with parameters
 - Literal-Block
- Idexable variable values
- Emacs Lisp evaluation of variables

v. Weaving

ID: `orgmode:gcr:vela:F71DD8BA-B853-4903-A348-400E13C0E6F8`

Help the reader make sense of the document by displaying it's internal properties.

```
(setq org-export-with-properties t)
```

- Stop your flow to monitor the export for errors
 - *<2016-01-19 Tue>* Expect it to start weaves for all weavers asynchronously. Does not do so; main thread is blocked until weaves complete.

```
(setq org-export-in-background nil)
```

Make sure that exported files are Unicode UTF-8.

```
(setq org-export-coding-system 'utf-8)
```

Do not preserve line-breaks when exporting instead let the destination format handle it as it sees fit. This doesn't work like I had expected and makes me wonder what I am confused about here. When I export to HTML text containing linebreaks no longer has linebreaks. This is what I expect. When I export that same text to a buffer though, the line breaks are included. Currently I use `sacha/unfill-paragraph` on that code.

```
(setq org-export-preserve-breaks nil)
```

When exporting anything, do not insert the exported content into the kill ring.

```
(setq org-export-copy-to-kill-ring nil)
```

By default I never want a table of contents generated. It is so easy to enable it with a property, it will be fine to turn it off.

```
(setq org-export-with-toc nil)
```

On export, maintain the literal spacing as found in the source block. Obviously this is important for make-files. It is really important everywhere because anything else would violate the law of least surprise.

```
(setq org-src-preserve-indentation t)
```

Maximize flexibility for weaving operations during export.

```
(setq org-export-allow-bind-keywords t)
```

Disable element caching because it might break weaves via this thread.

```
(setq org-element-use-cache nil)
```

A. exports

```
ID: orgmode:gcr:vela:57B3786B-017F-4F6E-89F9-05642304F3B6
```

Export code and/or results

Always share source blocks and their results. Whether or not to generate a result for a particular source block is configured per-block. If you don't want to share a result for a source block then disable storage of results on that block.

```
(help/set-org-babel-default-header-args :exports "both")
```

Use inline Source-Blocks to provide values read as part of the document. Don't show their source code. Allows inline Source-Blocks to function as *rich* macros when combined with `org-sbe`.

```
(help/set-org-babel-default-inline-header-args :exports "results")
```

B. wrap

```
ID: orgmode:gcr:vela:94D6B3BE-5DA1-499A-B5C7-A6B71710A1EA
```

Mark source block evaluation results

Inline-Source-Blocks are recognizable by their verbatim font. They do not interrupt the flow. Source-Blocks are their own entities. They stand out. Their results need to be visibly noticeably different for the reader by making them EXAMPLE special blocks.

```
(help/set-org-babel-default-header-args :wrap "EXAMPLE")
```

Diagramming languages require RESULTS output for exporting.

```
(help/set-org-babel-default-header-args:ditaa :wrap "RESULTS")
```

```
(help/set-org-babel-default-header-args:dot :wrap "RESULTS")
```

```
(help/set-org-babel-default-header-args:plantuml :wrap "RESULTS")
```

```
((:wrap . RESULTS) (:results . file) (:exports . results))
```