1. Org-Mode Exemple Complet Minimal

   ID: orgmode:gcr:vela:83868A6E-76C2-48AE-8A5A-6A3C97492321

   Configure EMACS to easily provide ECMs.

   Sysop is likely to use this often.

   Start EMACS with this command:

   ```
   open /Applications/Emacs.app -n --args --quick --load ~/src/help/.org-mode-ecm.emacs.el
   ```

   ```
   ;; -*- lexical-binding: t -*-
   ```

   (a) Principle of Least Astonishment (POLA)

   ID: orgmode:gcr:vela:43D9AB52-2157-4A1D-AD9C-D35996E91269

   i. Scoping Behavior

   ID: orgmode:gcr:vela:C0F8DDC4-C2B7-4578-B9B4-FA13E3BF0EBD

   EMACS values dynamic-scoping for now and in the future.
   EMACS values lexical-scoping in the future.
   There are pecuilar interactions between lexically and dynamically scope closures and special variables.
   With the future in mind, make the switch now.
   It is enabled with a non-nil buffer-local variable `lexical-binding`. The variable is inserted into each of the Web definitions that this system provides. Those definitions can't use Org-Mode comments so they are defined individually in each Web.

   ii. Load Behavior

   ID: orgmode:gcr:vela:B1A48CB2-9295-466E-947F-EE24D57DB548

   EMACS can load 3 different representations of a Emacs-Lisp source file code OOTB. The name of source code file is the value before the file extension. When you pass `load` a name it searches for an acceptable representation. Representation types are indicated by their extension name. `.el` is a human readable and uncompiled. `.elc` is not human readable and compiled. `.gz` is Gzip compressed and contains `.el` or `.elc` files.
   The variable `load-suffixes` determines the order for which text and byte-code representations are first searched by extension-name. The variable `load-file-rep-suffixes` determines the order for all other extension types.
   OOTB, EMACS combines the productivity of REPL style of development with the speed of compiled code by `load`'ing byte-code first, text second, and compressed third. This workflow gives you the fastest code at run-time and lets you experiment with new features stored in text. When you are ready to use them every time, you compile them. There is only one downside of this approach: when you forget that it works this way it can be confusing.
   When you forget about that style of system you end up with surprising behavior. The surprise comes when you forget to compile code and then write new code that depends on the now old version of that code. After the next build, your system can break in surprising ways. This violates the Principle of Least Astonishment.
   This system values predictability so it does the simplest thing possible: `load` searches for the newest representation of a file and uses that one. It assumes that Sysop has total and complete control over the management of file representations.
   This is the **first** thing that *ought* to happen in the initialization of **any** tangled system.

   ```
   (setq load-prefer-newer t)
   ```

   (b) Org-Mode Exemple Complet Minimal

   ID: orgmode:gcr:vela:57C69AB7-A317-4823-ABBF-7DE8A5E2151C

   Every new EMACS releases comes with the latest stable Org-Mode release. To get hot-fixes, cutting edge features, and easy patch creation though, you need to use the version from Git.

   These detailed and clear directions explain how ot run Org-Mode from Git. The only thing worth mentioning again is that in order to use **any** version of Org-Mode other than the one that comes OOTB you **must** load Org-Mode **before** anything else in your initialization file. It is easy to do! When you get unexpected Org-Mode behavior be sure to stop and investigate `org-version` and decide whether or not it is what you expect and prepare an ECM if necessary.

   Add the Org-Mode core distribution the load path.

```
(add-to-list 'load-path "~/src/org-mode/lisp")
```

Add the Org-Mode-Contributions distribution to the load path. The contributions are essential.

```
(add-to-list 'load-path "~/src/org-mode/contrib/lisp")
```

Allow single-character alphabetical bullet lists. This configuration must occur before loading Org-Mode. **Never** remove this from a submitted ECM.

```
(setq org-list-allow-alphabetical t)
```

Unchecked boxes prevent marking the parent as done. This configuration must occur before loading Org-Mode. **Never** remove this from a submitted ECM.

```
(setq org-enforce-todo-checkbox-dependencies t)
```

Use the Guillemet to demarcate source blocks. Reconfigure this yearly.

```
(setq org-babel-noweb-wrap-start "«")
(setq org-babel-noweb-wrap-end "»")
```

Load Org-Mode.

```
(require 'org)
```