The Org Manual

Release 9.3

The Org Mode Developers

This manual is for Org version 9.3.

目次

1 Introduction

1.1 Summary

Org is a mode モードです for keeping notes, maintaining TODO lists, and project planning with a fast and effective plain-text markup language. It also is an authoring system with unique support for literate programming and reproducible research.

Org is implemented 実装されています on top 上に of Outline モードの which これは makes it possible to keep the content of large files well structured. Visibility cycling and structure editing help to work with the tree. Tables are easily created with a built-in table editor. Plain text URL-like links connect つながります to websites, ウエブサイト、emails, Usenet messages, BBDB entries, and any files related to the projects.

Org develops organizational tasks around notes files that contain lists or information about projects as plain text. Project planning and task management make use 使用します of metadata which is part of an outline node. Based on this data, このデータに基いて specific entries can be extracted 抽出することができます in queries 問い合わせの中で and create 作成することができまますう dynamic 動的な*agenda views* を that also integrate the Emacs の calendar と diary も統合している Org Org-mode を使うことができます。can be used to implement 実装するために、many different project planning schemes, 多くの異なるプロジェクト警告スキームを such as David Allen の GTD システムといった、

Org files Org-mode のファイルは can serve 動くことができます as a single source authoring system with export to many different formats such as HTML やLaTeX や Open Document や Markdown といった New export backends can be derived 派生することもできますし from existing ones, or defined from scratch.

Org files can include 含むことができます source code blocks, ソースコードブロックを which これは makes Org uniquely suited for authoring technical documents with code examples. Org source code blocks are fully functional; they can be evaluated 評価することができます in place その場で and their results その結果を can be captured キャプチャすることができます in the file. そのファイルの中に This これは makes it possible 可能にします to create 作成することを a single file reproducible research compendium.

Org keeps simple things simple. When first fired up, 最初に立ち上げるときには it それは should feel 感じられるはずです like a straightforward, easy to use outliner. Complexity is not imposed, but a large amount of functionality is available when needed. 必用なときには Org is a toolbox. Org-mode は道具箱です。Many users actually run only a ---very personal ---fraction of Org's capabilities, and know that there is more whenever いつでも they need it.

All of this is achieved with strictly plain text files, the most portable and future-proof file format. Org runs in Emacs. Emacs is one of the most widely ported programs, so that Org-mode is available on every major platform.

There is a website for Org Org-mode 用のウエブサイトがあります which provides 提供しています links to the newest version of Org, as well as additional information, frequently asked questions (FAQ), links to tutorials, etc. このページは`https://orgmode.org` にあります。

An earlier version (7.3) of this manual is available as a paperback book from Network Theory Ltd. (`http://www.network-theory.co.uk/org/manual/`).

1.2 Installation

Org is included in all recent distributions of GNU Emacs, so you probably do not need to install it. それをインストールする必用はありませあ Most users will simply 単純に activate 有効にし Org Org-mode を and begin exploring its many features.

If, for one reason or another, you want to install Org on top of this pre-packaged version, there are three ways to do it:

- by using 使うことによって the Emacs パッケージシステムを
- by downloading Org as an archive; or
- by using 使うことによって Org's git repository. Org mode の Git レポジトリを

We **strongly recommend** sticking to a single installation method.

Using Emacs packaging system

Recent Emacs distributions include a packaging system パッケージシステムを which これは lets 可能にします you install インストールすることを Elisp のライブラリを You can install インストールすることができます Org with *M-x package-install RET org* を使って Org-mode を

> **Important:** You need to do this これを行なう必用があります in a session Emacs のセッションの中で where no '`.org`' file has been visited, i.e., すなわち where no Org built-in function have been loaded. Otherwise そうでなければ autoload Org functions will mess up the installation.

If you want to use 使いたいのであれば Org's package repository, check out the Org ELPA page (`https://orgmode.org/elpa.html`).

Downloading Org as an archive

You can download ダウンロードすることができます Org latest release Org-mode の最新のリリースを from Org's website (`https://orgmode.org/`) から In this case, この場合に make sure 確実にしてください you set the load-path correctly in your Emacs init file:

```
(add-to-list 'load-path "~/path/to/orgdir/lisp")
```

The downloaded archive contains 含んでいます contributed libraries 寄贈ライブラリを that are not included in Emacs. If you want to use 使いたいのであれば them, add the '`contrib/`' directory to your load-path:

```
(add-to-list 'load-path "~/path/to/orgdir/contrib/lisp" t)
```

Optionally, you can compile コンパイルして the files and/or install them in your system. Run '`make help`' を to list compilation and installation options.

Using Org's git repository

You can clone クローンすることができます Org's repository and install Org like this: このようにして

```
$ cd ~/src/
$ git clone git@code.orgmode.org:bzg/org-mode.git
$ cd org-mode/
$ make autoloads
```

Note 注意してください that in this case, この場合には`make autoloads` が is mandatory: 必須であることに it それは defines 定義しています Org's version in '`org-version.el`' and Org's autoloads in '`org-loaddefs.el`'.

Remember to add the correct load-path as described in the method above.

You can also compile コンパイルして with '`make`', generate 生成して the documentation with '`make doc`', create 作成して a local configuration with '`make config`' and install インストールすることができます Org with '`make install`'. Please どうか run 実行してください'`make help`' を to get the list of compilation/installation options.

For more detailed explanations on Org's build system, please どうか check チェックしてください the Org Build System page on Worg (`https://orgmode.org/worg/dev/org-build-system.html`).

1.3 Activation

Org-mode buffers need Font Lock to be turned on: オンになっていることが this is the default in Emacs

1.4 Feedback

If you find 見つけたり problems 問題を with Org, or if you have あれば questions, しつもん remarks, or ideas about it, please どうか send 送ってください an email E メールを to the Org mailing list `emacs-orgmode@gnu.org`. You can subscribe 購読することができます to the list from this web page (`https://lists.gnu.org/mailman/listinfo/emacs-orgmode`). If you are not a member of the mailing list, your mail will be passed to the list after a moderator has approved it[1].

For bug reports, バグレポートに対しては please どうか first 最初に try to reproduce 再生成しようとしてみてください the bug そのバグを with the latest version of Org available ---if you are running an outdated version, it is quite possible 可能性が極めて高いです that the bug そのバグが has been fixed already. If the bug そのバグが永続するのであれば、persists, prepare 準備して a report レポートを and provide as much information as possible, 可能なかぎり including 含む the version information of Emacs (*`M-x emacs-version`*) and Org (*`M-x org-version`*), as well as ともに the Org related setup in the Emacs init file. The easiest way to do this is to use the command

```
M-x org-submit-bug-report <RET>
```

which これは puts 入れます all this information into an Emacs mail buffer so that you only need to add 追加すればよいよういyour description. あなたの説明だけを If you あなたが are not sending 送信していないのであれば the Email E メールを from within Emacs, please どうか copy and paste コピーアンドペーストしてください the content この中身を into your Email program.

Sometimes ときどき you might face 出会うかもしれません a problem 問題点に due to an error in your Emacs or Org-mode setup. Before reporting a bug, it is very helpful to start Emacs with minimal customizations and reproduce the problem. Doing so often helps you determine 決定することを if the problem is with your customization or with Org-mode

[1] Please どうか consider 考慮してください subscribing 購読することを to the mailing list in order to minimize the work the mailing list moderators have to do.

itself. You can start はじめることができます a typical minimal session 典型的な最小限のセッションを with a command like the example below.

```
$ emacs -Q -l /path/to/minimal-org.el
```

However しかし if you are using 使っているのであれば Org-mode を as distributed 配布されている with Emacs といっしょに a minimal setup is not necessary. 必用ではありません In that case その場合には it is sufficient 十分です to start Emacs as '`emacs -Q`'. The '`minimal-org.el`' setup file can have 持つことができます contents as shown below. 下に示すように

```
;;; Minimal setup to load latest `org-mode'.

;; Activate debugging.
(setq debug-on-error t
      debug-on-signal nil
      debug-on-quit nil)

;; Add latest Org-mode to load path.
(add-to-list 'load-path (expand-file-name "/path/to/org-mode/lisp"))
(add-to-list 'load-path (expand-file-name "/path/to/org-mode/contrib/lisp" t))
```

If an error occurs, エラーが起きたら、a "backtrace" が can be very useful とても役に立つことがあります---see below 下を見てください。on how to create one. Often しばしば a small example file helps, 小さな例となるファイルが助けになります along with ともに clear information 明確な情報と about: 次についての

1. What exactly did you do?
2. What did you expect to happen?
3. かわりに何が起きたのか？

Thank you for helping to improve this program.

How to create a useful backtrace

If working with Org produces an error with a message メッセージがついた you do not understand, あなたが理解できない you あなたは may have hit a bug. バグに出会ったのかもしれません The best way to report this is by providing, 提供することによって in addition 加えて to what was mentioned above, a backtrace. This これは is information 情報です from the built-in debugger about where and how the error occurred. Here これは is how to produce a useful backtrace: 役に立つバックトレースを生成方法です:

1. Reload uncompiled versions of all Org-mode Lisp files. The backtrace contains 含んでいます much more information ずっと多くの情報を if it is produced with uncompiled code. To do this, これを行なうには、use 次を使ってください:

   ```
   C-u M-x org-reload <RET>
   ```

   or, from the menu: Org → Refresh/Reload → Reload Org uncompiled.
2. Then, activate the debugger:

   ```
   M-x toggle-debug-or-error <RET>
   ```

   or, from the menu: Options → Enter Debugger on Error.
3. Do whatever you have to do to hit the error. Do not forget to document the steps you take.

4. When you hit the error, a '`*Backtrace*`' buffer appears on the screen. Save 保存して this buffer to a file ---for example 例えば using *C-x C-w* を使って---and attach it to your bug report.

1.5 Typesetting Conventions Used in this Manual

TODO keywords, tags, properties, etc.

Org Org-mode は uses 使います various syntactical elements: TODO keywords, tags, property names, keywords, blocks, etc. などです In this manual we use the following conventions:

'`TODO`'
'`WAITING`' TODO keywords are written with all capitals, even if they are user-defined.

'`boss`'
'`ARCHIVE`' Tags are case-sensitive. User-defined tags are written in lowercase; built-in tags with special meaning are written as they should appear 出現するべき in the document, usually 通常は with all capitals.

'`Release`'
'`PRIORITY`'
User-defined properties are capitalized; built-in properties with special meaning are written with all capitals.

'`TITLE`'
'`BEGIN`' ... '`END`'
Keywords and blocks are written 書かれますが in uppercase 大文字で to enhance their readability, その読みやすさを高めるために but you can use lowercase 小文字を使うことができます。in your Org files. あなたの Org-mode のファイルの中では

Key bindings and commands

The manual lists both the keys and the corresponding commands for accessing a functionality. Org-mode は often 頻繁に uses 使います the same key 同じキーを for different functions, 異なる関数に対して depending on context. コンテキストに依存して The command このコマンドは that is bound to such keys has a generic name, like `org-metaright` のようんあ In the manual we will, wherever possible, 可能な場所ではどこでも give the function that is internally called by the generic command. For example, 例えば in the chapter on document structure, *M-RIGHT* は will be listed リストされます to call 呼び出すと`org-do-demote` を while 一方で in the chapter on tables, it will be listed to call `org-table-move-column-right`.

2 Document Structure

Org Org-mode は is based on Outline モードに基いていて、and provides 提供しています flexible commands 柔軟性のあるコマンドを to edit the structure of the document. ドキュメントの構造を編集するための

2.1 Outlines

Org is implemented on top of Outline mode. Outlines allow a document to be organized in a hierarchical structure, which, これは least for me, 少なくとも私いとっては is the best representation of notes and thoughts. ノートと思考の An overview of this structure is achieved by folding, i.e., すなわち hiding 隠すことによって large parts of the document to show only the general document structure and the parts currently being worked on. Org greatly simplifies the use of outlines by compressing the entire show and hide functionalities into a single command, `org-cycle`, これは which is bound バインドされています to the これは`TAB` キーにバインドされています。

2.2 Headlines

Headlines define the structure of an outline tree. The headlines in Org start with one or more stars, on the left margin[1]. For example:

```
* Top level headline
** Second level
*** Third level
    some text
*** Third level
    more text
* Another top level headline
```

The name defined in `org-footnote-section` is reserved. Do not use it as a title for your own headings.

Some people find the many stars too noisy and would prefer an outline that has white-space followed by a single star as headline starters. 節 15.9 [Clean View], ページ 248, を見てください。

Headline are not numbered. 見出しには番号はつけられません。However, しかし you may want to dynamically 動的に number some, or all, of them. 節 15.10 [Dynamic Headline Numbering], ページ 250, を見てください。

An empty line after the end of a subtree is considered みなされ part 一部であえうろ of it その and is hidden when the subtree is folded. However, しかし if you leave at least two empty lines, one empty line remains visible after folding the subtree, in order to structure the collapsed view. このふるまいを変更するには、変数`org-cycle-separator-lines` を見てください。

[1] See 見てください the variables 変数`org-special-ctrl-a/e` と`org-special-ctrl-k` と`org-ctrl-k-protect-subtree` を to configure 設定するために special behavior of *C-a*, *C-e*, and *C-k* in headlines. Note 注意してください also that clocking only works with headings indented less than 30 stars.

2.3 Visibility Cycling

2.3.1 Global and local cycling

Outlines アウトラインは make it possible 可能にします to hide 隠すことを parts 一部を of the text テキストお in the buffer. バッファの中の Org Org-mode は uses 使います just two commands, bound バインドされている to `TAB` and *`S-TAB`* to change the visibility in the buffer.

`TAB` (`org-cycle`)
: *Subtree cycling*: Rotate current subtree among the states

```
,-> FOLDED -> CHILDREN -> SUBTREE --.
'-----------------------------------'
```

Point must be on a headline for this to work[2].

`S-TAB` (`org-global-cycle`)
`C-u TAB`
: *Global cycling*: Rotate the entire buffer among the states

```
,-> OVERVIEW -> CONTENTS -> SHOW ALL --.
'--------------------------------------'
```

When *`S-TAB`* is called with a numeric prefix argument N, the CONTENTS view up to headlines of level N are shown. Note 注意してください that inside tables（章 3 [Tables], ページ 19, を見てください）、*`S-TAB`* は jumps to the previous field instead. かわりに

You can run 実行することができます global cycling グローバルな切り替えを using `TAB` を使って only if point is at the very beginning of the buffer, but not on a headline, and ˜org-cycle-global-at-bob ˜ is set to a non-`nil` value. `nil` でない値に

`C-u C-u TAB` (`org-set-startup-visibility`)
: Switch back to the startup visibility of the buffer（節 2.3.2 [Initial visibility], ページ 8, を見てください）。

`C-u C-u C-u TAB` (`outline-show-all`)
: Show all, including 含む、全てを表示します。drawers. ドロワーを

`C-c C-r` (`org-reveal`)
: Reveal context around point, showing the current entry, the following heading and the hierarchy above. Useful 役に立ちます for working 作業することに対して near 近くで a location that has been exposed by a sparse tree command（節 2.6 [Sparse Trees], ページ 12, を見てください）or an agenda command（節 10.5 [Agenda Commands], ページ 121, を見てください）。With a prefix argument 1 つの前置引数といっしょだと、show, 表示します on each level, all sibling headings. With a double prefix argument, also show the entire subtree of the parent.

`C-c C-k` (`outline-show-branches`)
: Expose all the headings of the subtree, CONTENTS view for just one subtree.

[2] See, however, しかし the option `org-cycle-emulate-tab`.

C-c TAB (`outline-show-children`)
: Expose all direct children of the subtree. With a numeric prefix argument N, expose all children down to level N.

C-c C-x b (`org-tree-to-indirect-buffer`)
: Show the current subtree in an indirect buffer[3]. With a numeric prefix argument, N, go up to level N and then take that tree. If N is negative N が負であれば then go up that many levels. With a *C-u* prefix, do not remove the previously used indirect buffer.

C-c C-x v (`org-copy-visible`)
: Copy the *visible* text in the region into the kill ring.

2.3.2 Initial visibility

When Emacs first visits an Org file, the global state is set to OVERVIEW に設定されます i.e., すなわち only the top level headlines are visible

2.3.3 Catching invisible edits

Sometimes ときどき you may inadvertently 間違って edit 編集してしまい an invisible part 不可視の部分を of the buffer バッファの and be confused on what has been edited and how to undo the mistake. Setting 設定するということが`org-catch-invisible-edits` を to non-`nil` `nil` でない値に helps 助けてくれます preventing this. これを防ぐことを See the docstring ドキュメント文字列を見てください。of this option このオプションの on how Org Org-mode が should catch 捉えて、invisible edits 不可視部分の and process them. それを処理する方法については、

2.4 Motion

The following commands jump to other headlines in the buffer.

C-c C-n (`outline-next-visible-heading`)
: Next heading.

C-c C-p (`outline-previous-visible-heading`)
: Previous heading.

C-c C-f (`org-forward-same-level`)
: Next heading same level.

C-c C-b (`org-backward-same-level`)
: Previous heading same level.

C-c C-u (`outline-up-heading`)
: Backward to higher level heading.

C-c C-j (`org-goto`)
: Jump ジャンプしま to a different place 異なる場所へ without changing 変更することなく the current outline visibility. 現在のアウトラインの可視性を Shows

[3] The indirect buffer インダイレクトバッファは contains 含んでいます the entire buffer, but is narrowed to the current tree. Editing the indirect buffer also changes the original buffer, but without affecting visibility in that buffer. For more information about indirect buffers, see 節 “Indirect Buffers” in `emacs`.

表示します the document structure ドキュメント構造を in a temporary buffer, 一時的なバッファの中で where そこでは you can use 使うことができます the following keys 次のキーを to find your destination: 目的地を見るけるために

`TAB`	Cycle visibility.
`DOWN` / `UP`	Next/previous visible headline.
`RET`	Select this location.
`/`	Do a Sparse-tree search

The following keys work if you turn off `org-goto-auto-isearch`

n / *p*	Next/previous visible headline.
f / *b*	Next/previous headline same level.
u	One level up.
0 ... *9*	Digit argument.
q	Quit.

変数`org-goto-interface` も見てください。

2.5 Structure Editing

M-RET (`org-meta-return`)

Insert 挿入します a new heading, item or row.

If the command このコマンドが is used 使われたら at the *beginning* はじめで of a line, 行の and if there is a heading or a plain list item（節 2.7 [Plain Lists], ページ 13, を見てください）at point, ポイントに the new heading/item is created 作成されます*before* the current line. When used 使われるときには at the beginning はじめで of a regular line 通常の行の of text, テキストの turn that line into a heading.

When this command is used in the middle 真ん中で of a line, the line is split and the rest of the line becomes なります the new item or headline. If you do not want 望まないのであれば the line to be split, この行が分割されることを customize `org-M-RET-may-split-line` をカスタマイズしてください

Calling the command with a *C-u* prefix unconditionally 無条件で inserts 挿入します a new heading at the end of the current subtree, thus よって preserving its contents. With a double *C-u C-u*< prefix, the new heading is created 作成されます at the end 終わりに of the parent subtree 親のサブツリーの instead. かわりに

C-RET (`org-insert-heading-respect-content`)

Insert 挿入します a new heading at the end of the current subtree.

M-S-RET (`org-insert-todo-heading`)

Insert 挿入します。new TODO entry TODO エントリーを with same level as current heading. 現在の見出しと同じレベルの変数`org-treat-insert-todo-heading-as-state-change` も見てください。

C-S-RET (`org-insert-todo-heading-respect-content`)

Insert 挿入します new TODO entry with same level as current heading. Like *C-RET* と同じように the new headline 新しい見出しは is inserted after the current subtree.

TAB (org-cycle)
: In a new entry with no text yet, the first TAB demotes レベルを下げます the entry to become a child of the previous one. The next TAB makes it a parent, and so on, all the way to top level. Yet another TAB, and you are back to the initial level.

M-LEFT (org-do-promote)
: Promote current heading by one level.

M-RIGHT (org-do-demote)
: Demote current heading by one level.

M-S-LEFT (org-promote-subtree)
: Promote the current subtree by one level.

M-S-RIGHT (org-demote-subtree)
: Demote the current subtree by one level.

M-UP (org-move-subtree-up)
: Move subtree up, サブツリーを上に移動します: i.e., すなわち swap 入れ替えます with previous subtree of same level.

M-DOWN (org-move-subtree-down)
: Move subtree down, サブツリーを下に移動します: i.e., すなわち swap 入れかえます with next subtree of same level.

C-c @ (org-mark-subtree)
: Mark the subtree at point. ポイントにあるサブツリーをマークします。Hitting repeatedly marks subsequent subtrees of the same level as the marked subtree.

C-c C-x C-w (org-cut-subtree)
: Kill subtree, i.e., すなわち remove it from buffer but save in kill ring. With a numeric prefix argument N, kill N sequential subtrees.

C-c C-x M-w (org-copy-subtree)
: Copy subtree to kill ring. With a numeric prefix argument N, copy the N sequential subtrees.

C-c C-x C-y (org-paste-subtree)
: Yank subtree from kill ring. This これは does modify 変更します the level of the subtree to make sure the tree fits in nicely at the yank position. The yank level can also be specified 指定することもできます with a numeric prefix argument, or by yanking after a headline marker like '****'.

C-y (org-yank)
: Depending 依存して on the 変数org-yank-adjusted-subtrees とorg-yank-folded-subtrees に Org's internal yank command pastes subtrees folded and in a clever way, using the same command as *C-c C-x C-y*. With the default settings, no level adjustment takes place, but the yanked tree is folded unless doing so そうすることが would swallow text previously visible. Any prefix argument to this command forces 強制します a normal yank to be executed, 実行されることを with the prefix passed along. A good way to force a normal yank is *C-u C-y*. If you use yank-pop after a yank, it yanks previous kill items plainly, without adjustment and folding.

C-c C-x c (`org-clone-subtree-with-time-shift`)
: Clone a subtree by making a number of sibling copies of it. You are prompted for the number of copies to make, and you can also specify 指定することもできます if any timestamps in the entry should be shifted. シフトされるべき This これが can be useful, 役に立つことがあります for example, 例えば to create 作成するために a number of tasks 複数のタスクを related 関係する to a series of lectures to prepare. For more details, さらに詳細についてはsee 見てください the docstring ドキュメント文字列を of the コマンドorg-clone-subtree-with-time-shift の

C-c C-w (`org-refile`)
: Refile 再ファイルします entry or region to a different location. 節 9.5 [Refile and Copy], ページ 104, を見てください。

C-c ^ (`org-sort`)
: Sort same-level entries. When there is an active region, all entries in the region are sorted. Otherwise そうでなければ the children 子が of the current headline 現在の見出しの are sorted. ソートされます The command このコマンドは prompts プロンプトを出します for the sorting method, ソート方法を求めて which これは can be alphabetically, numerically, by time---first timestamp with active preferred, creation time, scheduled time, deadline time---by priority, by TODO keyword---in the sequence the keywords have been defined in the setup---or by the value of a property. 逆順のソートも可能です。You can also supply 提供することもできます your own function to extract the sorting key. With a *C-u* prefix, sorting is case-sensitive. ソートは大文字と小文字を区別します。

C-x n s (`org-narrow-to-subtree`)
: Narrow buffer to current subtree.

C-x n b (`org-narrow-to-block`)
: Narrow buffer to current block.

C-x n w (`widen`)
: Widen buffer to remove narrowing.

*C-c ** (`org-toggle-heading`)
: Turn a normal line or plain list item into a headline---so that it becomes a subheading at its location. Also また turn a headline into a normal line by removing the stars. If there is an active region, turn all lines in the region into headlines. If the first line in the region was an item, turn only the item lines into headlines. Finally, 最後に if the first line is a headline, remove the stars from all headlines in the region.

When there is an active region---i.e., すなわち when Transient Mark mode is active---promotion and demotion work on all headlines in the region. To select 選択するには a region リージョンを of headlines, it is best to place both point and mark at the beginning of a line, mark at the beginning of the first headline, and point at the line just after the last headline to change. Note 注意してください that when point ポイントが is inside a table 表の内側にあるときには（章 3 [Tables], ページ 19, を見てください）、the Meta-Cursor キーが have 持つということに different functionality. 異なる機能を

2.6 Sparse Trees

An important feature of Org-mode is the ability to construct *sparse trees* for selected information in an outline tree, so that the entire document is folded 折り畳まれあす as much as possible, 可能なかぎり多く but the selected information is made visible 可視にされます along with ともに the headline structure above it[4]. Just try it out and you will see immediately how it works.

Org-mode は contains 含んでいます several commands creating such trees, all these commands can be accessed アクセスすることができます through a dispatcher: ディスパッチャをとおして

C-c / (`org-sparse-tree`)
: This これは prompts プロンプトを出します for an extra key to select a sparse-tree creating command.

C-c / r or *C-c / /* (`org-occur`)
: Prompts プロンプトを出します for a regexp 正規表現を求めて and shows 表示します a sparse tree スパースツリーを with all matches. 全てのマッチがついた If the match is in a headline, the headline is made visible. If the match is in the body of an entry, headline and body are made visible. In order to provide minimal context, also the full hierarchy of headlines above the match is shown, as well as the headline following the match. Each match is also highlighted; the highlights disappear when the buffer is changed by an editing command, or by pressing *C-c C-c* を押すことににによって[5]. When called 呼ばれるときには with a *C-u* prefix argument, previous highlights are kept, so several calls to this command can be stacked. 積み重なることがあります

M-g n or *M-g M-n* (`next-error`)
: Jump ジャンプします to the next sparse tree match 次のスパースツリーのマッチへ in this buffer. このバッファの中の

M-g p or *M-g M-p* (`previous-error`)
: Jump ジャンプしまづ to the previous sparse tree match 前のスパースツリーのマッチへ in this buffer. このバッファの中の

For frequently used sparse trees of specific search strings, you can use 使うことができます the 変数`org-agenda-custom-commands` を to define 定義するために fast keyboard access to specific sparse trees. These commands will then be accessible through the agenda dispatcher（節 10.2 [Agenda Dispatcher], ページ 108, を見てください）。例です:

```
(setq org-agenda-custom-commands
      '(("f" occur-tree "FIXME")))
```

defines 定義します the key *f* というキーを as a shortcut ショートカットとして for creating 作成するたんめの a sparse tree スパースツリーを matching マッチする the string '`FIXME`' という文字列に

The other sparse tree commands select headings based 基いて on TODO キーワード、tags, or properties and are discussed later 後のほうで in this manual. このマニュアルの

[4] See 見てください also the variable 変数`org-show-context-detail` もあ to decide how much context is shown around each match.

[5] This depends on the option `org-remove-highlights-with-change`.

To print a sparse tree, スパースツリーをプリントするには、you can use the Emacs のコマンド`ps-print-buffer-with-faces` を使うことができます。which does not print invisible parts of the document. ドキュメントの不可視の部分をプリントしない、Or または you can use 使うことができます the command コマンド`C-c C-e v` を to export エクスポートして only the visible part of the document ドキュメントの可視の部分だけを and print プリントするために the resulting file. 結果のファイルを

2.7 Plain Lists

Within an entry of the outline tree, hand-formatted lists can provide 提供することができあますう additional structure. 追加の構造を They also provide a way 方法も to create lists of checkboxes（節 5.6 [Checkboxes], ページ 59, を見てください）。Org Org-mode は supports サポートしています editing 編集を such lists, そのようなリストの and every exporter 全てのエクスポーターが（章 12 [Exporting], ページ 145, を見てください）can parse パースして and format フォーマットすることができまますう them.

Org knows ordered lists, unordered lists, and description lists.

- *Unordered* list items start with '`-`', '`+`', or '`*`'[6]

  as bullets. ビュレットとして

- *Ordered* list items start with a numeral followed by either a period or a right parenthesis[7], such as '`1.`' や'`1)`' といった[8] If you want 望むのであれば a list to start with a different value ---例えば 20 ---start はじめてください the text of the item with '`[@20]`'[9]. Those constructs これらの文法を can be used 使うことができます in any item of the list リストのいずれのアイテムの中でも in order to enforce 強制するために、a particular numbering. 特定の番号づけを

- *Description* list items are unordered list items, and contain the separator '`::`' to distinguish the description *term* from the description.

Items belonging to the same list must have the same indentation on the first line. In particular, if an ordered list reaches number '`10.`', then the 2-digit numbers must be written left-aligned with the other numbers in the list. An item ends before the next line that is less or equally indented than its bullet/number.

A list ends whenever いつでも every item has ended, which これは means 意味します before any line less or equally indented than items at top level. It also ends before two blank lines. In that case, その場合には all items are closed. これは例です:

```
* Lord of the Rings
```

[6] When using 使うときには'`*`' を as a bullet, lines must be indented so that they are not interpreted as headlines. Also, また when you are hiding leading stars to get a clean outline view, plain list items starting with a star may be hard to distinguish 区別することが from true headlines. In short: even though '`*`' is supported, it may be better to not use it for plain list items.

[7] You can filter out any of them by configuring 設定することによって`org-plain-list-ordered-item-terminator` を

[8] You can also get '`a.`', '`A.`', '`a)`' and '`A)`' by configuring `org-list-allow-alphabetical` を設定することによって To minimize 最小限にするために confusion with normal text, those are limited to one character only. Beyond that limit, そのリミットを超えると bullets ビュレットは automatically 自動的に become numbers.

[9] If there's a checkbox in the item, the cookie must be put *before* the checkbox. If you have activated alphabetical lists, you can also use counters like '`[@b]`'.

```
My favorite scenes are (in this order)
1. The attack of the Rohirrim
2. Eowyn's fight with the witch king
   + this was already my favorite scene in the book
   + I really like Miranda Otto.
3. Peter Jackson being shot by Legolas
   - on DVD only
   He makes a really funny face when it happens.
But in the end, no individual scenes matter but the film as a whole.
Important actors in this film are:
- Elijah Wood :: He plays Frodo
- Sean Astin :: He plays Sam, Frodo's friend. I still remember him
     very well from his role as Mikey Walsh in /The Goonies/.
```

Org supports サポートしています these lists これらのリストを by tuning filling and wrapping commands to deal with them correctly, and by exporting them properly（章 12 [Exporting], ページ 145, を見てください）。Since indentation is what governs the structure of these lists, many structural constructs like '`#+BEGIN_`' blocks can be indented インデントすることができます to signal 示すために that they それが belong to a particular item.

If you find that using a different bullet for a sub-list ---than that ビュレットではなく used for the current list-level ---improves readability, customize カスタマイズしてください the 変数`org-list-demote-modify-bullet` を To get 得るに a greater difference of indentation between items and theirs sub-items, customize カスタマイズしてください`org-list-indent-offset` を

The following commands act on items when point ポイントが is in the first line 最初の行おの中にあるときには of an item アイテムの---the line 行です with the bullet or number. ビュレットか数字がある Some of them imply the application of automatic rules to keep list structure intact. If some of these actions これらのアクションのいくつかが get in your way, あなたの邪魔をするのであふるえば、configure 設定してください`org-list-automatic-rules` を to disable 無効にするには them そららを individually. 個々に

`TAB` (`org-cycle`)

> Items アイテムを can be folded 折り畳むことができます just like headline levels. 見出しのレベルとちょうど同じように Normally 通常は this これは works 動きます only if point ポイントが is on a plain list item. プレインリストアイテムの上にある場合にのに For more details, さらに詳細については see the 変数`org-cycle-include-plain-lists` を見てください。
>
> If this variable この変数が is set 設定されいる場合には to `integrate` に plain list items プレインリストアイテムは are treated 扱われます like low-level headlines. The level of an item アイテムのレベルは is then given 与えられます by the indentation インデント量によって of the bullet/number. ビュレット/数字の Items アイテムは are always 常に subordinate 配下になります to real headlines, however; しかし the hierarchies remain completely separated. In a new item with no text yet, まだテキストがない the first `TAB` demotes レベルを下げます the item to become a child of the previous one. Subsequent `TAB`s move the item to meaningful levels in the list and eventually get it back to its initial position.

M-RET (`org-insert-heading`)
: Insert 挿入します new item at current level. With a prefix argument, 1 つの前置引数といっしょだと、force 強制します a new heading（節 2.5 [Structure Editing], ページ 9, を見てください）。If this command is used in the middle of an item, that item is *split* in two, and the second part becomes the new item[10]. If this command is executed *before item's body*, the new item is created 作られます*before* the current one.

M-S-RET
: Insert 挿入します a new item with a checkbox（節 5.6 [Checkboxes], ページ 59, を見てください）。

S-UP
S-DOWN
: Jump ジャンプしまづ to the previous/next item in the current list, but only if `org-support-shift-select` is off オフの場合のみです[11]. If not, そうでなくても you can still それでも use 使うことができますあ paragraph jumping commands like *C-UP* や*C-DOWN* のような to quite similar effect.

M-UP
M-DOWN
: Move 移動します the item including 含む subitems up/down[12], i.e., すなわち swap with previous/next item of same indentation. If the list is ordered, renumbering is automatic.

M-LEFT
M-RIGHT
: Decrease/increase the indentation of an item, leaving children alone.

M-S-LEFT
M-S-RIGHT
: Decrease/increase the indentation of the item, including 含む subitems. Initially, the item tree is selected based 基いて on current indentation. When these commands are executed several times in direct succession, the initially selected region is used, even if the new indentation would imply a different hierarchy. To use the new hierarchy, 新しい階層構造を使うには、break the command chain by moving point. ポイントを移動することによって

  As a special case, using this command on the very first item of a list moves the whole list. This behavior このふるまいを can be disabled 無効にすることができあます。by configuring 設定することによって、`org-list-automatic-rules` を The global indentation of a list has no influence on the text *after* the list.

C-c C-c
: If there is a checkbox（節 5.6 [Checkboxes], ページ 59, を見てください）in the item line, toggle the state of the checkbox. In any case, verify 確認します bullets and ビュレットと indentation consistency インデントの一貫性を in the whole list. リスト全体の

C-c -
: Cycle the entire list level through the different itemize/enumerate bullets (‘`-`’, ‘`+`’, ‘`*`’, ‘`1.`’, ‘`1)`’) or a subset of them, depending 依存して、on `org-plain-list-ordered-item-terminator` に the type of list, and its indentation. With

[10] If you do not want the item to be split, customize カスタマイズしてください the variable 変数`org-M-RET-may-split-line` を

[11] If you want to cycle around items that way, you may customize カスタマイズすることができます`org-list-use-circular-motion` を

[12] See `org-list-use-circular-motion` for a cyclic behavior.

a numeric prefix argument N, 数値の前置引数 N といっしょだと、select the Nth bullet from this list. If there is an active region アクティブなリージョンがあれば when calling this, selected text is changed into an item. With a prefix argument, 1 つの前置引数といっしょだしたお、all lines are converted to list items. If the first line already was a list item, any item marker is removed from the list. Finally, 最後に even without an active region, アクティブなリージョンがないとしても a normal line is converted into a list item. リストアイテムへ変換されます

*C-c **

Turn 変えます a plain list item into a headline ---so that it becomes a subheading at its location. 詳細な説明については、節 2.5 [Structure Editing], ページ 9, を見てください。

*C-c C-**

Turn the whole plain list into a subtree of the current heading. Checkboxes（節 5.6 [Checkboxes], ページ 59, を見てください）become TODO, respectively DONE, keywords when unchecked, respectively checked.

S-LEFT
S-RIGHT

This command このコマンドは also cycles bullet styles ビュレットおスタイルを when point is in on the bullet or anywhere どこかに in an item line, details 詳細は depending 依存して on `org-support-shift-select` に

C-c ^

Sort the plain list. Prompt for the sorting method: numerically, alphabetically, by time, or by custom function.

2.8 Drawers

Sometimes ときどき you want to keep information associated with an entry, but you normally 通常は do not want to see it. それを見たくはない For this, このためには、Org-mode には has *drawers* があります They それは can contain 含んむことができますあ anything 何でも but a headline 見出しと and another drawer. ドロワーはこのように見えます:

```
** This is a headline
Still outside the drawer
:DRAWERNAME:
This is inside the drawer.
:END:
After the drawer.
```

You can interactively insert インタラクティブに挿入することができます a drawer ドロワーを at point ポイントの場所に by calling `org-insert-drawer` を呼びだすことによって which これは is bound バインドされています to *C-c C-x d* い With an active region, アクティブなリージョンがあれば、this command puts the region inside the drawer. With a prefix argument, this command calls 呼出しますorg-insert-property-drawer を which これは creates 作成します a ‘PROPERTIES’ ドロワーを right below すぐ下に the current headline. Org-mode は uses 使います this special drawer この特別なドロワーを for storing 保存するために properties プロパティを（章 7 [Properties and Columns], ページ 66, を見てください）。You cannot use 使うことはできません it それを for anything else.

ドロワーのキーワードに対する補完も、*M-TAB* を使って可能です[13]。

Visibility cycling 可視性の切り替えが（節 2.3 [Visibility Cycling], ページ 7, を見てください）on the headline hides 隠して and shows 表示します the entry, そのエントリーを but keep the drawer collapsed to a single line. 1 行に In order to look inside the drawer, ドロワーの内側を見るためには、you need to move 移動する必用があります point to the drawer line ポイントをドロワー行に and press there. そこで`TAB` を押す必用があります。

You can also arrange アレンジすることもできます for state change notes 状態の変更のノートと（節 5.3.2 [Tracking TODO state changes], ページ 56, を見てください）and clock times（節 8.4 [Clocking Work Time], ページ 82, を見てください）to be stored 保存されるように in a '`LOGBOOK`' ドロワーの中に If you want to store 保存したいのであれば a quick note there, in a similar way to state changes, use 次を使ってください:

C-c C-z Add 追加します a time-stamped note to the '`LOGBOOK`' drawer.

2.9 Blocks

Org-mode は uses 使います'`#+BEGIN`' ... '`#+END`' ブロックを for various purposes from including 含む source code examples（節 11.6 [Literal Examples], ページ 141, を見てください）to capturing time logging information（節 8.4 [Clocking Work Time], ページ 82, を見てください）。These blocks can be folded 折り畳んで and unfolded 拡げることができますう by pressing `TAB` を押すことににによって in the '`#+BEGIN`' 行の中で You can also get all blocks 全てのブロックを folded 折り畳んでおくこともできあますう at startup スタートアップ時に by configuring 設定することによって the variable 変数`org-hide-block-startup` を or on a per-file basis ファイル毎には by using 使うことによって

```
#+STARTUP: hideblocks
#+STARTUP: nohideblocks
```

2.10 Creating Footnotes

Org-mode は supports the creation of footnotes. 脚注の作成をサポートしています。

A footnote is started by a footnote marker in square brackets in column 0, no indentation allowed. インデントは許可されていません It それは ends at the next footnote definition, headline, or after two consecutive empty lines. The footnote reference is simply 単純に the marker in square brackets, inside text. Markers マーカーは always 常に start はじまります with '`fn:`' で例です:

```
The Org homepage[fn:1] now looks a lot better than it used to.
...
[fn:1] The link is: https://orgmode.org
```

Org-mode extends the number-based syntax to *named* footnotes and optional inline definition. Here これらは are the valid references: 有効な参照です

'`[fn:NAME]`'
: A named footnote reference, where *NAME* is a unique label word, or, for simplicity of automatic creation, a number.

[13] Many desktops intercept *M-TAB* を to switch windows. ウインドウを切り替えるために Use 使ってください*C-M-i* か*ESC TAB* を instead. かわりに

‘`[fn:: This is the inline definition of this footnote]`’
: A LaTeX-like anonymous footnote where the definition is given 与えられる directly 直接 at the reference point.

‘`[fn:NAME: a definition]`’
: An inline definition of a footnote, which これは also また specifies 指定します a name 名前も for the note. Since Org allows 許可しているので multiple references 複数の参照を to the same note, you can then use 使うことができます‘`[fn:NAME]`’ を to create 作成するために additional references. 追加の参照を

Footnote labels 脚注のラベルを can be created 作成することもできますし automatically, 自動的に、or you can create 作成することもできます names 名前を yourself. あなた自身で This これは is handled 扱われます by the 変数`org-footnote-auto-label` と and its corresponding ‘`STARTUP`’ keywords. 詳細については、その変数のドキュメント文字列を見てください。

The following command handles footnotes:

C-c C-x f The footnote action command.

When point ポイントが is on a footnote reference, 脚注の参照にあるときには jump ジャンププします to the definition. その定義へ When it is at a definition, jump to the ---first ---reference.

Otherwise, そうでなければ create 索引します a new footnote. 新しい脚注を Depending 依存して on the 変数`org-footnote-define-inline`[14] に the definition is placed right into the text as part of the reference, or separately into the location determined 決定される by the 変数`org-footnote-section` によって

When this command is called このコマンドが呼ばれるときには with a prefix argument, 1 つの前置引数といっしょに a menu of additional options is offered: 追加のオプションのメニューが提供されあしもう:

s	Sort the footnote definitions by reference sequence.
r	Renumber the simple ‘`fn:N`’ footnotes.
S	Short for first *r*, then *s* action.
n	Rename all footnotes into a ‘`fn:1`’ ... ‘`fn:n`’ sequence.
d	Delete the footnote at point, including definition and references.

Depending 依存して、on the 変数`org-footnote-auto-adjust`[15] に renumbering and sorting footnotes can be automatic 自動にすることができまう after each insertion or deletion.

C-c C-c If point ポイントが is on a footnote reference, 脚注の参照の上にあるときんいは jump ジャンプします to the definition. その定義へ If it それが is at the definition, 定義にあれば jump ジャンプします back to the reference. When called at a footnote location 脚注の場所で呼ばれるときには、with a prefix argument, 1 つの前置引数といっしょに offer the same menu as *C-c C-x f* と同じメニューを提供します。

[14] The corresponding in-buffer setting is: ‘`#+STARTUP: fninline`’ or ‘`#+STARTUP: nofninline`’.

[15] The corresponding in-buffer options are ‘`#+STARTUP: fnadjust`’ and ‘`#+STARTUP: nofnadjust`’.

C-c C-o or *mouse-1/2*

Footnote labels 脚注のラベルは are also links リンクでもあするいます to the corresponding definition 対応する定義か or reference, 参照に対する and you can use the usual commands 通常のコマンドを使うことができます to follow these links. これらのリンクをたどるために

3 Tables

Org comes with ついてきます a fast and intuitive table editor. Spreadsheet-like calculations are supported using the Emacs Calc package（`calc` を見てください）。

3.1 Built-in Table Editor

Org makes it easy to format tables in plain ASCII. Any line with '`|`' as the first non-whitespace character is considered part of a table. '`|`' is also the column separator[1]. Moreover, a line starting with '`|-`' is a horizontal rule. It separates rows explicitely. Rows before the first horizontal rule are header lines. 表はこのように見えますっ

```
| Name  | Phone | Age |
|-------+-------+-----|
| Peter |  1234 |  17 |
| Anna  |  4321 |  25 |
```

A table is re-aligned 再整列するえます automatically 自動的に each time 毎回 you press 押すときには`TAB` か`RET` か*`C-c C-c`* を inside the table. 表の内側で`TAB` は also moves to the next field —`RET` は to the next row ---and creates 作成します new table rows 新しい表の行を at the end 終わりに of the table or before horizontal lines. The indentation of the table is set by the first line. Horizontal rules are automatically expanded 自動的に展開されます on every re-align to span the whole table width. So, to create the above table, you would only type

```
|Name|Phone|Age|
|-
```

and then press `TAB` to align the table and start filling in fields. Even faster would be to type '`|Name|Phone|Age`' followed by *`C-c RET`*.

When typing text into a field, Org treats *`DEL`*, *`Backspace`*, and all character keys in a special way, so that inserting and deleting avoids 避けるように shifting other fields. Also, また when typing *immediately* after point was moved into a new field with `TAB`, *`S-TAB`* or `RET`, the field is automatically 自動的に made blank. 空にされます If this behavior このふるまいが is too unpredictable あまりにも予期できないことであれば for you, あなたにとって configure 設定してください the option オプション`org-table-auto-blank-field` を

Creation and conversion

`C-c` `|` (`org-table-create-or-convert-from-region`)

Convert the active region アクティブなリージョンを to table. If every line contains 含んでいたら at least one `TAB` character, the function assumes that the material is tab separated. If every line contains 含んでいたら a comma, コンマを comma-separated values (CSV) are assumed. 仮定されます If not, そうでなければ、lines are split 分割されます at whitespace into fields. 空白でフィールドに You can use 使うことができます a prefix argument 前置引数を to force 強制するために a specific separator: 特定のセパレータを*`C-u`* は forces 強制します CSV を*`C-u C-u`* は forces 強制します`TAB` を*`C-u C-u C-u`* は prompts プロンプトを出します for a regular expression to match the separator, and a numeric

[1] To insert 挿入するには a vertical bar into a table field, use '`\vert`' or, inside a word '`abc\vert{}def`'.

argument N indicates that at least N consecutive spaces, or alternatively a `TAB` will be the separator.

If there is no active region, アクティブなリージョンがあれば、this command このコマンドは creates 作成します an empty Org table. 空の Org-mode の表を But しかし it is easier just to start typing, like *| N a m e | P h o n e | A g e RET | - TAB*.

Re-aligning and field motion

C-c C-c (`org-table-align`)
: Re-align the table 表を再整列します。without moving point. ポイントを移動することなく

`TAB` (`org-table-next-field`)
: Re-align the table, 表を再整列し、move to the next field. 次のフィールドへ移動します。Creates 作成します。a new row 新しい行を if necessary. 必用であれば

C-c SPC (`org-table-blank-field`)
: Blank the field at point.

S-TAB (`org-table-previous-field`)
: Re-align, move to previous field.

`RET` (`org-table-next-row`)
: Re-align the table 表を再整列し and move down to next row. 次の行へ移送します。Creates 作成します。a new row 新しい行を if necessary. 必用であれば新しい行を At the beginning はじめか or end of a line, `RET` が still inserts 挿入します a new line, 改行を挿入するおで so it それを can be used 使うことができます to split a table. 表を分割するために

M-a (`org-table-beginning-of-field`)
: Move 移動します to beginning of the current table field, or on to the previous field.

M-e (`org-table-end-of-field`)
: Move 移動します to end of the current table field, or on to the next field.

Column and row editing

M-LEFT (`org-table-move-column-left`)
: Move 移動します the current column left. 左に

M-RIGHT (`org-table-move-column-right`)
: Move 移動します the current column right. 右い

M-S-LEFT (`org-table-delete-column`)
: Kill the current column.

M-S-RIGHT (`org-table-insert-column`)
: Insert 挿入します。a new column 新しい列を to the left 左側に of point position. ポイントの場所の

M-UP (`org-table-move-row-up`)
: Move 移動します the current row up. 上に

M-DOWN (`org-table-move-row-down`)
: Move 移動します the current row down. 下へ

M-S-UP (`org-table-kill-row`)
: Kill the current row or horizontal line.

S-UP (`org-table-move-cell-up`)
: Move 移動します cell up 上へ by swapping 入れかえることによって with adjacent cell.

S-DOWN (`org-table-move-cell-down`)
: Move 移動します cell down by swapping 入れかえることによって with adjacent cell.

S-LEFT (`org-table-move-cell-left`)
: Move 移動します cell left 左へ by swapping 入れかえることによって with adjacent cell. 隣接するセルと

S-RIGHT (`org-table-move-cell-right`)
: Move 移動します cell right by swapping 入れかえることによって with adjacent cell.

M-S-DOWN (`org-table-insert-row`)
: Insert 挿入します a new row above the current row. With a prefix argument, 1 つの前置引数といっしょだと、the line この行は is created 作られます below the current one.

C-c - (`org-table-insert-hline`)
: Insert 挿入します a horizontal line 水平線を below current row. With a prefix argument, 1 つの前置引数といっしょだと the line is created 作成されます above the current line.

C-c RET (`org-table-hline-and-move`)
: Insert 挿入します a horizontal line 水平線を below current row, and move point into the row below that line.

C-c ^ (`org-table-sort-lines`)
: Sort the table lines in the region. The position of point indicates the column to be used for sorting, and the range of lines is the range between the nearest horizontal separator lines, or the entire table. If point ポイントが is before the first column, you are prompted プロンプトが出あますう for the sorting column. If there is an active region, the mark マークは specifies 指定します the first line and the sorting column, while point ポイントは should be in the last line to be included into the sorting. The command このコマンドは prompts プロンプトを出します for the sorting type, alphabetically, numerically, or by time. You can sort ソートすることができます in normal or reverse order. You can also supply 提供することもできあますう your own key extraction and comparison functions. When called with a prefix argument, 1 つの前置引数といっしょに呼ばれるときには、alphabetic sorting アルファベット順のソートは is case-sensitive. 大文字と小文字を区別します。

Regions

C-c C-x M-w (`org-table-copy-region`)
: Copy a rectangular region from a table to a special clipboard. Point and mark determine edge fields of the rectangle. If there is no active region, copy just the current field. The process ignores horizontal separator lines.

C-c C-x C-w (`org-table-cut-region`)
: Copy a rectangular region from a table to a special clipboard, and blank all fields in the rectangle. So this is the "cut" operation.

C-c C-x C-y (`org-table-paste-rectangle`)
: Paste a rectangular region into a table. The upper left corner ends up in the current field. All involved fields are overwritten. If the rectangle does not fit into the present table, the table is enlarged as needed. 必用に応じて The process ignores horizontal separator lines.

M-RET (`org-table-wrap-region`)
: Split the current field at point position and move the rest to the line below. If there is an active region, and both point and mark are in the same column, the text in the column is wrapped to minimum width for the given number of lines. A numeric prefix argument may be used 使うことができあますう to change the number of desired lines. If there is no region, but you specify a prefix argument, the current field is made blank, and the content is appended to the field above.

Calculations

C-c + (`org-table-sum`)
: Sum the numbers in the current column, or in the rectangle defined by the active region. The result is shown in the echo area and can be inserted 挿入することができます with *C-y* を使って

S-RET (`org-table-copy-down`)
: When current field is empty, 現在のフィールドが空であれば、copy コピーします from first non-empty field above. 上にある When not empty, 空でないときには、copy コピーします current field down to next row and move point along with ともに it. それと Depending 依存して、on the 変数`org-table-copy-increment` に integer field values can be incremented インクリメントすることができあす during copy. コピーの間に Integers that are too large are not incremented, however. しかし Also, また a `0` prefix argument temporarily disables the increment. This key is also used by shift-selection and related modes（節 15.13.2 [Conflicts], ページ 252, を見てください）。

Miscellaneous

*C-c `* (`org-table-edit-field`)
: Edit the current field in a separate window. This is useful これが役に立ちます for fields that are not fully visible（節 3.2 [Column Width and Alignment], ページ 23, を見てください）。When called 呼ばれるときには with a *C-u* 前置引数といっしょに just 単に make the full field visible, 可視にします so that it can be edited 編集することができる in place. その場で When called 呼ばれる

ときには with two 2 つの*C-u* 前置引数といっしょに make the editor window 変数ウインドウに follow point through the table and always 常に show 表示しまますう the current field. 現在のフィールドを The follow mode exits 終了します automatically 自動的に when point leaves the table, or when you repeat this command with *C-u C-u C-c `*.

M-x org-table-import
: Import a file as a table. The table この表は should be TAB かスペースで分けれれているべきです。あ Use, 使ってください for example, 例えば to import a spreadsheet table or data from a database, because these programs generally can write 書くことができます TAB-separated text files. This command works 動きます by inserting 挿入することによって the file into the buffer and then converting the region to a table. Any prefix argument is passed on to the converter, which これは uses 使います it それを to determine 決定するために the separator. セパレータを

C-c | (`org-table-create-or-convert-from-region`)
: Tables can also be imported by pasting tabular text into the Org buffer, selecting the pasted text with *C-x C-x* and then using the *C-c |* command（[Creation and conversion], ページ 19, を見てください）。

M-x org-table-export
: Export エクスポートします the table, 表を by default デフォルトで as a TAB で分けられたファイルとして Use 使ってください for data exchange データの交換に対して with, for example, 例えば spreadsheet スプレッドシートや or database programs. データベースプログラムとの The format フォーマットは used to export the file can be configured 設定することができます in the variable 変数`org-table-export-default-format` の中で You may also use 使うこともできます properties プロパティ =TABLE_EXPORT_FILE= と‘`TABLE_EXPORT_FORMAT`’ を to specify 指定するために the file name and the format for table export in a subtree. Org supports サポートしています quite general formats for exported tables. The exporter format is the same as the format used by Orgtbl radio tables, see 節 A.5.3 [Translator functions], ページ 263, for a detailed description.

3.2 Column Width and Alignment

The width of columns is automatically 自動的に determined 決定されま by the table editor. The alignment of a column is determined automatically 自動的に from the fraction 割合から of number-like versus non-number fields in the column.

Editing a field may modify 変更することがあります alignment of the table. Moving a contiguous row or column---i.e., すなわち using `TAB` か`RET` を---automatically 自動的に再整列します re-aligns it. それを If you want to disable 無効にしたいのであれば this behavior, このふるまいを set `org-table-automatic-realign` を`nil` に設定してください。In any case, いずれの場合でも、you can always 常に align 揃えることができます manually 手動で a table: 表を

C-c C-c (`org-table-align`)
: Align the current table.

Setting the option `org-startup-align-all-tables` re-aligns all tables in a file upon visiting it. You can also set 設定することもできます this option このオプションを on a per-file basis ファイルごとに with:

```
#+STARTUP: align
#+STARTUP: noalign
```

Sometimes ときどき a single field 1 つのフィールドや or a few fields 複数のフィールドが need to carry 持つ必用があり more text, より多くのテキストを leading 導きます to inconveniently wide columns. 不便なほど幅の広い列に Maybe たぶん you want to hide away several columns or display them with a fixed width, regardless of content, as shown in the following example.

```
|---+----------------------+--------|           |---+-------...|...|
|   | <6>                  |        |           |   | <6>   ...|...|
| 1 | one                  | some   |   ----\   | 1 | one   ...|...|
| 2 | two                  | boring |   ----/   | 2 | two   ...|...|
| 3 | This is a long text | column |           | 3 | This i...|...|
|---+----------------------+--------|           |---+-------...|...|
```

To set the width of a column, 列の幅を設定するには、one field anywhere どこかにある 1 つのフィールドが、in the column この列の中の may contain 含むことができます just the string '`<N>`' where *N* specifies 指定します the width 幅を as a number of characters. 文字数として You control コントロールします displayed width of columns with the following tools:

C-c TAB (`org-table-toggle-column-width`)
: Shrink or expand current column.

  If a width cookie specifies 指定します a width W for the column, shrinking it displays 表示します the first W visible characters only. Otherwise, そうでなければ the column この列は is shrunk 縮められます to a single character. 1 文字にん

  When called 呼ばれるときには before the first column 最初の列の前か or after the last one, ask for a list of column ranges to operate on.

C-u C-c TAB (`org-table-shrink`)
: Shrink all columns with a column width. Expand the others.

C-u C-u C-c TAB (`org-table-expand`)
: Expand all columns.

To see 見るには the full text 完全な適切を of a shrunk field, 縮められているフィールドの hold the mouse over it: a tool-tip window then shows 表示します the full contents of the field. Alternatively, あるいは、*C-h .* (`display-local-help`) が reveals them, too. For convenience, 便利なために any change 全ての変更が near the shrunk part of a column expands it.

Setting the option `org-startup-shrink-all-tables` shrinks all columns containing 含んでいる a width cookie 幅のクッキーを in a file the moment it is visited. You can also set 設定することができます this option このオプションを on a per-file basis ファイルごとに with:

```
#+STARTUP: shrink
```

If you would like to overrule 上書きしたいのであれば the automatic alignment 自動的な揃えを of number-rich columns to the right 右へ、and of string-rich columns to the left, 左へという you can use 使うことができます‘`<r>`’, ‘`<c>`’ か‘`<l>`’ を in a similar fashion. You may also combine 組み合わせることもできまますう alignment and field width like this: このようにして‘`<r10>`’.

Lines which only contain these formatting cookies are removed automatically 自動的に取り除かれます upon exporting the document.

3.3 Column Groups

When Org exports tables, Org-mode が表をエクスポートするときには it does so そうします by default デフォルトで without vertical lines 垂直線なしで because that is visually more satisfying in general. 一般的に Occasionally ときおり however, しかし vertical lines 垂直線が can be useful 役に立つことがあります to structure 構造化するために a table 表を into groups of columns, 列のグループに much like ちょうど同じように horizontal lines can do for groups of rows. In order to specify column groups, 列のグループを指定するには、you can use 使うことができます a special row 特別な行を where the first field 最初のフィールドが contains only ‘`/`’ だけを含んでいる The further fields can either contain ‘`<`’ を to indicate 示すための that this column should start はじめるべきだ a group, ‘`>`’ to indicate the end of a column, or ‘`<>`’ (no space between ‘`<`’ and ‘`>`’) to make a column a group of its own. Upon export, boundaries between column groups are marked with vertical lines. これは例です: これは例です:

```
| N | N^2 | N^3 | N^4 | sqrt(n) | sqrt[4](N) |
|---+-----+-----+-----+---------+------------|
| / |  <  |     |  >  |       < |          > |
| 1 |  1  |  1  |  1  |       1 |          1 |
| 2 |  4  |  8  | 16  |  1.4142 |     1.1892 |
| 3 |  9  | 27  | 81  |  1.7321 |     1.3161 |
|---+-----+-----+-----+---------+------------|
#+TBLFM: $2=$1^2::$3=$1^3::$4=$1^4::$5=sqrt($1)::$6=sqrt(sqrt(($1)))
```

It is also sufficient to just insert the column group starters after every vertical line you would like to have:

```
| N | N^2 | N^3 | N^4 | sqrt(n) | sqrt[4](N) |
|---+-----+-----+-----+---------+------------|
| / | <   |     |     | <       |            |
```

3.4 The Orgtbl Minor Mode

If you like 好きであれば the intuitive way 直感的な方法が the Org table editor works, 動く you might also want to use it in other modes like Text mode or Mail mode. The minor mode マイナーモード Orgtbl mode が makes this possible. これを可能にしています You can always 常に toggle トグルすることができまますう the mode このモードを with *M-x orgtbl-mode* を使って To turn it on by default, デフォルトでそれをオンにするには、for example 例えば in Message mode, use

```
(add-hook 'message-mode-hook 'turn-on-orgtbl)
```

Furthermore, さらに with some special setup, it is possible 可能です to maintain tables in arbitrary syntax with Orgtbl mode. For example, 例えば it is possible 可能です to

construct 構築することがLaTeX tables with the underlying ease and power of Orgtbl mode, including 含む spreadsheet capabilities. スプレッドシート機能を For details, 詳細については、see 節 A.5 [Tables in Arbitrary Syntax], ページ 260, を見てください。

3.5 The Spreadsheet

The table editor makes use of the Emacs Calc package to implement spreadsheet-like capabilities. It can also evaluate Emacs Lisp forms to derive fields from other fields. While fully featured, Org's implementation is not identical to other spreadsheets. For example, 例えば Org Org-mode は knows 知っています the concept of a *column formula* that will be applied to all non-header fields in a column 1 つの列の中にある without having to copy コピーする必用なく the formula 数式を to each relevant field. There is also a formula debugger, and a formula editor with features for highlighting fields in the table corresponding to the references at point in the formula, moving these references by arrow keys.

3.5.1 References

To compute 計算するには fields フィールドを in the table 表の中の from other fields, 他のフィールドから formulas 吸うしきいは must reference 参照しなければなりません other fields or ranges. In Org, fields can be referenced by name, by absolute coordinates, and by relative coordinates. 相対座標によって To find 見つけだすには out what the coordinates of a field are, press 押してください`C-c ?` を in that field, or press `C-c }` to toggle the display of a grid.

Field references

Formulas can reference 参照することげんえきます the value of another field in two ways. Like in any other spreadsheet, you may reference 参照することができます fields フィールドを with a letter/number combination like '`B3`' のような meaning 意味する the second field in the third row. However, しかし Org prefers 好みます to use 使うことを another, 別の、more general representation that looks like this: このように見える

Range references

You may reference 参照することができます a rectangular range 矩形領域を of fields フィールドん by specifying 指定することによって two field references 2 つのフィールド参照を connected つなげられた by two dots '`..`'. If both fields are in the current row, you may simply 単純に use 使うことができますが'`$2..$7`' を but if at least one field is in a different row, you need to use 使う必用がありますthe general 一般的な'`@ROW$COLUMN`' フォーマットをあ at least for the first field, i.e., すなわち the reference must start with '`@`' in order to be interpreted correctly. Examples:

'`$1..$3`'	first three fields in the current row
'`$P..$Q`'	range, using column names（節 3.5.10 [Advanced features], ページ 36, を見てください）
'`$<<<..$>>`'	start in third column, continue to the last but one
'`@2$1..@4$3`'	six fields between these two fields (same as '`A2..C4`')
'`@-1$-2..@-1`'	3 fields in the row above, starting from 2 columns on the left
'`@I..II`'	between first and second hline, short for '`@I..@II`'

Range references return a vector of values that can be fed into Calc vector functions. Empty fields in ranges are normally 通常は suppressed, 抑圧されます so that the vector contains

含んでいます only the non-empty fields. For other options with the mode switches '`E`', '`N`' and examples, see 節 3.5.2 [Formula syntax for Calc], ページ 28.

Field coordinates in formulas

One of the very first actions during evaluation of Calc formulas and Lisp formulas is to substitute '`@#`' and '`$#`' in the formula with the row or column number of the field where the current result will go to. The traditional Lisp formula equivalents are `org-table-current-dline` and `org-table-current-column`. Examples:

'`if(@# % 2, $#, string(""))`'
: Insert 挿入します column number on odd rows, set field to empty on even rows.

'`$2 = '(identity remote(FOO, @@#$1))`'
: Copy text or values of each row of column 1 of the table named *FOO* into column 2 of the current table.

'`@3 = 2 * remote(FOO, @@1$$#)`'
: Insert 挿入します the doubled value of each column of row 1 of the table named *FOO* into row 3 of the current table.

For the second and third examples, table *FOO* must have at least 少なくとも as many rows or columns as the current table. Note 注意してください that this これが is inefficient[2] for large number of rows.

Named references

'`$name`' is interpreted as the name of a column, parameter or constant. Constants are defined 定義されます globally グローバルに through the variable 変数`org-table-formula-constants` を and locally ローカルに---for the file ファイルに対して---through a line 行をとおして like this example: この例のような

```
#+CONSTANTS: c=299792458. pi=3.14 eps=2.4e-6
```

Also, また properties プロパティを（章 7 [Properties and Columns], ページ 66, を見てください）can be used 使うことができます as constants 定数として in table formulas: 表の数式の中の for a property プロパティ'`Xyz`' に対して use the name '`$PROP_Xyz`', and the property will be searched in the current outline entry and in the hierarchy above it. If you have the '`constants.el`' package, it will also be used to resolve constants, including 含む natural constants like '`$h`' for Planck's constant, and units like '`$km`' for kilometers[3]. Column names and parameters can be specified in special table lines. These are described below, see 節 3.5.10 [Advanced features], ページ 36. All names must start はじまらなければなりません with a letter, and further consist of letters and numbers.

Remote references

You may also reference 参照することもできます constants, 定数 fields and ranges from a different table, either in the current file or even in a different file. The syntax is

```
remote(NAME,REF)
```

[2] The computation time scales as O(N^2) because table *FOO* is parsed for each field to be copied.

[3] The file '`constants.el`' can supply the values of constants in two different unit systems, '`SI`' and '`cgs`'. Which one is used depends on the value of the variable 変数`constants-unit-system` の You can use 使うことができます the '`STARTUP`' オプション'`constSI`' and '`constcgs`' を to set 設定するために this value for the current buffer.

where *NAME* can be the name of a table in the current file as set by a '`#+NAME:`' line before the table. It can also be the ID of an entry, even in a different file, and the reference then refers to the first table in that entry. *REF* is an absolute field or range reference as described above for example 例えば'`@3$3`' や'`$somename`' のような valid in the referenced table.

When *NAME* に has あるときには the format '`@ROW$COLUMN`', it is substituted with the name or ID found in this field of the current table. For example '`remote($1, @@>$2)`' ⇒ '`remote(year_2013, @@>$1)`'. The format '`B3`' is not supported because it can not be distinguished from a plain table name or ID.

3.5.2 Formula syntax for Calc

A formula can be any algebraic expression understood by the Emacs Calc package. Note 注意してください that Calc has the non-standard convention that '`/`' has lower precedence than '`*`', so that '`a/b*c`' is interpreted as '`(a/(b*c))`'. Before evaluation by `calc-eval`（節 "Calling Calc from Your Programs" in `calc` を見てください）variable substitution 変数の置き換えは takes place 起きます according 従って to the rules ルールに described above. 上で説明されている

The range vectors 範囲ベクトルを can be directly 直接 fed 与えることができます into the Calc vector functions like `vmean` と`vsum` のような

A formula can contain an optional mode string after a semicolon. This string consists of flags to influence Calc and other modes during execution. By default, デフォルトで Org Org-mode は uses 使います the standard Calc modes (precision 12, angular units degrees, fraction and symbolic modes off). The display format, however, しかし has been changed 変更されています to '`(float 8)`' へ to keep tables 表を compact. コンパクトに保つために The default settings デフォルト設定を can be configured 設定することができます using the 変数`org-calc-default-modes` を使って

'`p20`' Set the internal Calc calculation precision to 20 digits.

'`n3`', '`s3`', '`e2`', '`f4`'
Normal, scientific, engineering or fixed format of the result of Calc passed back to Org. Calc formatting is unlimited in precision as long as the Calc calculation precision is greater.

'`D`', '`R`' Degree and radian angle modes of Calc.

'`F`', '`S`' Fraction and symbolic modes of Calc.

'`T`', '`t`', '`U`' Duration computations in Calc or Lisp, 節 3.5.4 [Durations and time values], ページ 30.

'`E`' If and how to consider empty fields. Without '`E`' なしでは empty fields in range references are suppressed so that the Calc vector or Lisp list contains 含んでいます only the non-empty fields. With '`E`' the empty fields are kept. For empty fields in ranges or empty field references the value '`nan`' (not a number) is used in Calc formulas and the empty string is used for Lisp formulas. Add '`N`' を to use 0 を使うには instead かわりに for both formula types. For the value of a field the mode '`N`' has higher precedence than '`E`'.

‘N’ Interpret all fields as numbers, use 0 for non-numbers. See 見てください the next section 次のセクションを to see how this is essential for computations with Lisp formulas. In Calc formulas it is used only occasionally because there number strings are already interpreted as numbers without ‘N’. なしで

‘L’ Literal, for Lisp formulas only. 次のセクションを見てください。

Unless you use large integer numbers or high-precision calculation and display for floating point numbers you may alternatively provide a `printf` format specifier to reformat the Calc result after it それが has been passed back to Org instead かわりに of letting Calc already do the formatting[4]. A few examples:

‘$1+$2’	Sum of first and second field
‘$1+$2;%.2f’	Same, format result to two decimals
‘exp($2)+exp($1)’	数学関数を使うことができる
‘$0;%.1f’	Reformat current cell to 1 decimal
‘($3-32)*5/9’	Degrees F → C conversion
‘$c/$1/$cm’	Hz → cm conversion, using ‘constants.el’
‘tan($1);Dp3s1’	Compute in degrees, precision 3, display SCI 1
‘sin($1);Dp3%.1e’	Same, but use `printf` specifier for display
‘vmean($2..$7)’	Compute column range mean, using vector function
‘vmean($2..$7);EN’	Same, but treat empty fields as 0
‘taylor($3,x=7,2)’	Taylor series of $3, at x=7, second degree

Calc は also contains 含んでいます a complete set of logical operations（節 “Logical Operations” in `calc` を見てください）For example 例えば

‘if($1 < 20, teen, string(""))’
‘"teen"’ if age ‘$1’ is less than 20, else the Org table result field is set to empty with the empty string.

‘if("$1" == "nan" || "$2" == "nan", string(""), $1 + $2); E f-1’
Sum of the first two columns. When at least 少なくとも one of the input fields is empty 空のときには the Org table result field is set to empty. ‘E’ is required to not convert empty fields to 0. ‘f-1’ is an optional Calc format string similar to ‘%.1f’ but leaves empty results empty.

‘if(typeof(vmean($1..$7)) == 12, string(""), vmean($1..$7); E’
Mean value of a range unless there is any empty field. Every field in the range that is empty is replaced by ‘nan’ which lets 可能にします‘vmean’ result in ‘nan’. Then ‘typeof =’ 12= detects the ‘nan’ from `vmean` and the Org table result field is set to empty. Use 使ってください this これを when the sample set is expected to never have missing values.

‘if("$1..$7" == "[]", string(""), vmean($1..$7))’
Mean value of a range with empty fields skipped. Every field in the range that is empty is skipped. When all fields 全てのフィールドが in the range are empty the mean value is not defined and the Org table result field is set to empty. Use

[4] The printf reformatting is limited in precision because the value passed to it is converted into an "integer" or "double". The "integer" is limited in size by truncating the signed value to 32 bits. The "double" is limited in precision to 64 bits overall which leaves approximately 16 significant decimal digits.

使ってください this これを when the sample set サンプルの集合が can have 持つことができるときに a variable size. 可変のサイズを

‘`vmean($1..$7); EN`’
: To complete the example before: Mean value of a range with empty fields counting as samples with value 0. Use 使ってください this これを only when incomplete sample sets should be padded パディングされるべき with 0 to the full size.

You can add 追加することができます your own Calc functions defined 定義される in Emacs Lisp with `defmath` and use them それを in formula syntax for Calc.

3.5.3 Emacs Lisp forms as formulas

It is also possible to write a formula 数式を書くことも可能です。in Emacs Lisp で This can be これが useful 役に立つことがあります for string manipulation and control structures, if Calc's functionality is not enough.

If a formula starts with a single-quote followed by an opening parenthesis, then it is evaluated as a Lisp form. The evaluation should return 返すべき either a string or a number. Just as with Calc formulas, you can specify 指定することができます modes and a `printf` format after a semicolon.

With Emacs Lisp のフォームを使う場合には、you need to be conscious 意識しておく必用があります about the way 方法に field references are interpolated into the form. By default, デフォルトで a reference is interpolated as a Lisp string (in double-quotes) containing 含んでいる the field. そのフィールドを If you provide the ‘`N`’ mode switch, all referenced elements are numbers ---non-number fields will be zero---and interpolated as Lisp numbers, without quotes. クオートなしで If you provide the ‘`L`’ flag, all fields are interpolated literally, そのま without quotes. クオートなしで For example, 例えば if you want a reference to be interpreted as a string by the Lisp form, enclose the reference operator itself in double-quotes, like ‘`"$3"`’. Ranges are inserted as space-separated fields, so you can embed them in list or vector syntax.

Here これらは are a few examples ---note how the ‘`N`’ mode is used when we do computations in Lisp:

‘`'(concat (substring $1 1 2) (substring $1 0 1) (substring $1 2))`’
: Swap the first two characters of the content of column 1.

‘`'(+ $1 $2);N`’
: Add columns 1 and 2, equivalent to Calc's ‘`$1+$2`’.

‘`'(apply '+ '($1..$4));N`’
: Compute the sum of columns 1 to 4, like Calc's ‘`vsum($1..$4)`’.

3.5.4 Durations and time values

If you want to compute time values use the ‘`T`’, ‘`t`’, or ‘`U`’ flag, either in Calc formulas or Elisp formulas:

```
|  Task 1 |   Task 2 |    Total |
|---------+----------+----------|
|    2:12 |     1:47 | 03:59:00 |
```

```
|    2:12 |     1:47 |    03:59 |
| 3:02:20 | -2:07:00 |     0.92 |
#+TBLFM: @2$3=$1+$2;T::@3$3=$1+$2;U::@4$3=$1+$2;t
```

Input duration values must be of the form '`HH:MM[:SS]`', where そこでは seconds are optional. 秒はオプションです With the '`T`' flag, computed durations are displayed 表示されます as '`HH:MM:SS`' として（上の最初の数式を見てください）With the '`U`' flag, seconds are omitted so that the result is only '`HH:MM`'（上の 2 番目の数式を見てください）Zero-padding of the hours field depends 依存します upon the value 値に of the 変数`org-table-duration-hour-zero-padding` の

With the '`t`' flag, computed durations are displayed according 従って to the value 値に of the option `org-table-duration-custom-format` の which これは defaults デフォルトで to `hours` であり、and displays 表示します the result 結果を as a fraction of hours（上の例の 3 番目の数式を見てください）。

Negative duration values can be manipulated as well, and integers are considered as seconds in addition and subtraction.

3.5.5 Field and range formulas

To assign 割り当てるにぬうあ a formula 数式を to a particular field, 特定のフィールドに対して type タイプしてください it それを directly 直接 into the field, フィールドの中に preceded 前につけて by '`:=`', for example 例えば'`vsum(@II..III)`' のようにして When you press 押すときには`TAB` or `RET` or *`C-c C-c`* を with point ポイントを still in the field, そのフィールドの中に置いたままで the formula is stored as the formula for this field, evaluated, and the current field is replaced with the result.

Formulas are stored in a special '`TBLFM`' keyword located directly below the table. その表の直接下にある If you type the equation in the fourth field of the third data line in the table, the formula looks like '`@3$4=$1+$2`'. When inserting/deleting/swapping column and rows 列と行を with the appropriate commands, *absolute references* (but not relative ones) in stored formulas are modified in order to still reference the same field. To avoid this from happening, in particular in range references, anchor ranges at the table borders (using '`@<`', '`@>`', '`$<`', '`$>`'), or at hlines using the '`@I`' notation. Automatic adaptation of field references does not happen 起きません if you edit the table structure with normal editing commands ---you must fix 修正しなければなりませわい the formulas yourself.

Instead え of typing an equation 数式をタイプするかわりに、into the field, フィールドの中へ you may also use the following command 次のコマンドを使うこともできます:

`C-u C-c =` (`org-table-eval-formula`)
: Install a new formula for the current field. The command このコマンドは prompts プロンプトを出します for a formula with default taken from the '`TBLFM`' keyword, applies it to the current field, and stores it.

The left-hand side of a formula can also be a special expression in order to assign the formula to a number of different fields. There is no keyboard shortcut to enter such range formulas. To add them, それらを追加するには use 使ってください the formula editor（節 3.5.8 [Editing and debugging formulas], ページ 33, を見てください）or edit the '`TBLFM`' keyword directly.

'`$2=`'
: Column formula, valid for the entire column. This is so common that Org treats these formulas in a special way, see 節 3.5.6 [Column formulas], ページ 32.

‘`@3=`’ Row formula, applies to all fields in the specified row. ‘`@>=`’ means the last row.

‘`@1$2..@4$3=`’
Range formula, applies to all fields in the given rectangular range. This can also be used to assign a formula to some but not all fields in a row.

‘`$NAME=`’ Named field, see 節 3.5.10 [Advanced features], ページ 36.

3.5.6 Column formulas

When you assign 割り当てるときいは a formula 数式を to a simple column reference like ‘`$3=`’, the same formula is used in all fields of that column, with the following very convenient exceptions: (i) If the table contains 含んでいます horizontal separator hlines with rows above and below, everything before the first such hline is considered part of the table *header* and is not modified by column formulas. Therefore a header is mandatory when you use column formulas and want to add hlines to group rows, like for example to separate a total row at the bottom from the summand rows above. (ii) Fields that already get a value from a field/range formula are left alone by column formulas. These conditions make column formulas very easy to use.

To assign 割り当てるには a formula 数式を to a column, type it directly into any field in the column, preceded by an equal sign, like ‘`=$1+$2`’. When you press 押すときには`TAB` か`RET` か*C-c C-c* を with point ポイントを still in the field, そのフィールドの中に置いたままで the formula is stored as the formula for the current column, evaluated and the current field replaced with the result. If the field contains 含んでいます only ‘`=`’, the previously stored formula for this column is used. For each column, Org only remembers the most recently used formula. In the ‘`TBLFM`’ keyword, column formulas look like ‘`$4=$1+$2`’. The left-hand side of a column formula can not be the name of column, it must be the numeric column reference or ‘`$>`’.

Instead of typing an equation 数式をタイプするかわりに、into the field, フィールドの中に you may also use the following command: 次のコマンドを使うこともできま:

C-c = (`org-table-eval-formula`)
Install a new formula for the current column and replace current field with the result of the formula. The command このコマンドは prompts プロンプトを出します for a formula, 数式を求めて with default taken from the ‘`TBLFM`’ keyword, applies it to the current field and stores it. With a numeric prefix argument, e.g., 例えば*C-5 C-c =*, the command applies it to that many consecutive fields in the current column.

3.5.7 Lookup functions

Org has three predefined Emacs Lisp functions for lookups in tables.

‘`(org-lookup-first VAL S-LIST R-LIST &optional PREDICATE)`’
Searches for the first element *S* in list *S-LIST* for which

```
(PREDICATE VAL S)
```

is non-`nil`; `nil` でない値である returns 返します the value from the corresponding position in list *R-LIST*. デフォルトの*PREDICATE* は`equal` です。Note 注意してください that the parameters *VAL* and *S* are passed to *PREDICATE* in

the same order as the corresponding parameters are in the call to `org-lookup-first`, where *VAL* precedes *S-LIST*. If *R-LIST* is `nil`, the matching element *S* of *S-LIST* is returned.

‘`(org-lookup-last VAL S-LIST R-LIST &optional PREDICATE)`’
Similar to `org-lookup-first` above, but searches for the *last* element for which *PREDICATE* が is non-`nil`. `nil` でない値である

‘`(org-lookup-all VAL S-LIST R-LIST &optional PREDICATE)`’
Similar to `org-lookup-first` に似ていますが、but searches 検索します for *all* elements for which *PREDICATE* が is non-`nil`, `nil` でない値である and returns *all* corresponding values. This function can not be used by itself in a formula, because it returns a list of values. However, しかし powerful lookups 強力なルックアップを can be built when this function is combined with other Emacs Lisp functions.

If the ranges used in these functions contain empty fields, the ‘`E`’ モードが mode for the formula should usually 通常は be specified: 指定されるべきです: otherwise そうでなければ empty fields 空のフィールドは are not included 含められません in *S-LIST* and/or *R-LIST* which can, これが for example, 例えば result in an incorrect mapping from an element of *S-LIST* to the corresponding element of *R-LIST*.

These three functions これら 3 つの関数を can be used 使うことができます to implement 実装するために associative arrays, count matching cells, rank results, group data, etc. For practical examples 実践的な例については、see this tutorial on Worg (`https://orgmode.org/worg/org-tutorials/org-lookups.html`) を見てください。

3.5.8 Editing and debugging formulas

You can edit 編集することができます individual formulas in the minibuffer or directly in the field. Org can also prepare a special buffer with all active formulas of a table. When offering 提供するときには a formula 数式を for editing, 編集用に Org Org-mode は converts 変換します references to the standard format (like ‘`B3`’ or ‘`D&`’) if possible. 可能であれば If you prefer 好んのであれば to only work with the internal format（=@3$2= や‘`$4`’ のような）configure 設定してください。the 変数`org-table-use-standard-references` を

C-c = or *C-u C-c* = (`org-table-eval-formula`)
Edit the formula associated with the current column/field in the minibuffer. 節 3.5.6 [Column formulas], ページ 32, と節 3.5.5 [Field and range formulas], ページ 31, を見てください。

C-u C-u C-c = (`org-table-eval-formula`)
Re-insert the active formula (either a field formula, or a column formula) into the current field, so that you can edit it directly in the field. The advantage 利点は、over editing in the minibuffer ミニバッファの中での編集にん対する is that you can use the コマンド*C-c ?* を使うことができるということです

C-c ? (`org-table-field-info`)
While editing a formula in a table field, highlight the field(s) referenced by the reference at point position in the formula.

C-c } (`org-table-toggle-coordinate-overlays`)
: Toggle トグルします the display of row and column numbers for a table, using overlays. These are updated each time the table is aligned; you can force it with *C-c C-c*.

C-c { (`org-table-toggle-formula-debugger`)
: Toggle トグルします the formula debugger on and off. 下を見てください。

C-c ' (`org-table-edit-formulas`)
: Edit all formulas for the current table in a special buffer, where the formulas are displayed one per line. If the current field has an active formula, point in the formula editor marks it. While inside the special buffer, Org automatically 自動的に highlights ハイライトします any field or range reference at point position. You may edit, 編集することができます remove and add formulas, and use 使うことができます the following commands:

  C-c C-c or *C-x C-s* (`org-table-fedit-finish`)
  : Exit the formula editor and store the modified formulas. With *C-u* prefix, also apply the new formulas to the entire table.

  C-c C-q (`org-table-fedit-abort`)
  : Exit the formula editor 数式エディタを終了します。without installing changes. 変更をインストールするこたんあく

  C-c C-r (`org-table-fedit-toggle-ref-type`)
  : Toggle トグルします all references in the formula editor between standard (like '`B3`') and internal (like '`@3$2`').

  `TAB` (`org-table-fedit-lisp-indent`)
  : Pretty-print or indent Lisp formula at point. When in a line 行の中にいるときには containing 含んでいる a Lisp の数式を format the formula according 従って to Emacs Lisp のルールに Another `TAB` collapses the formula back again. In the open formula, `TAB` re-indents just like in Emacs Lisp mode.

  M-TAB (`lisp-complete-symbol`)
  : Complete Lisp symbols, just like in Emacs Lisp mode.

  S-UP, *S-DOWN*, *S-LEFT*, *S-RIGHT*
  : Shift the reference at point. For example, 例えば if the reference is '`B3`' and you press *S-RIGHT*, it becomes '`C3`'. This also works for relative references and for hline references.

  M-S-UP (`org-table-fedit-line-up`)
  : Move 移動します the test line for column formulas up in the Org buffer.

  M-S-DOWN (`org-table-fedit-line-down`)
  : Move 移動します the test line for column formulas down in the Org buffer.

  M-UP (`org-table-fedit-scroll-up`)
  : Scroll up the window displaying the table.

M-DOWN (`org-table-fedit-scroll-down`)
: Scroll down the window displaying the table.

C-c }
: Turn the coordinate grid in the table on and off.

Making a table field blank does not remove the formula associated with the field, because that is stored in a different line---the '`TBLFM`' keyword line. During the next recalculation, the field will be filled again. To remove 取り除くには a formula from a field, you have to give an empty reply when prompted for the formula, or to edit the '`TBLFM`' keyword.

You may edit 編集することができます the '`TBLFM`' keyword directly and re-apply the changed equations with *C-c C-c* in that line or with the normal recalculation commands in the table.

Using multiple '`TBLFM`' lines

You may apply the formula temporarily. This is useful これが役に立ちます when you want to switch the formula applied to the table. Place multiple '`TBLFM`' keywords right after the table, and then press *C-c C-c* on the formula to apply. これは例です:

```
| x | y |
|---+---|
| 1 |   |
| 2 |   |
#+TBLFM: $2=$1*1
#+TBLFM: $2=$1*2
```

Pressing *C-c C-c* in the line of '`#+TBLFM: $2=$1*2`' yields:

```
| x | y |
|---+---|
| 1 | 2 |
| 2 | 4 |
#+TBLFM: $2=$1*1
#+TBLFM: $2=$1*2
```

If you recalculate this table, with *C-u C-c ** を使って for example, 例えば you get the following result from applying only the first '`TBLFM`' keyword.

```
| x | y |
|---+---|
| 1 | 1 |
| 2 | 2 |
#+TBLFM: $2=$1*1
#+TBLFM: $2=$1*2
```

Debugging formulas

When the evaluation 評価が of a formula 数式の leads to an error, エラーに導くときには the field content そのフィールドの中身は becomes なります the string '`#ERROR`'. If you would like to see what is going on during variable substitution 変数の置き換えと and calculation in order to find 見つけるために a bug, バグを turn on formula debugging in the Tbl menu and repeat the calculation, for example 例えば by pressing *C-u C-u C-c = RET* を押すことににによって in a field. フィールドの中で詳細な情報が表示されます。

3.5.9 Updating the table

Recalculation 再計算は of a table 表の is normally 通常は not automatic, 自動でなはく、but needs to be triggered トリガーがかけられる必用があります by a command. コマンドによって To make recalculation 再計算を at least 少なくとも semi-automatic, 半自動にするには see 節 3.5.10 [Advanced features], ページ 36, を見てください

In order to recalculate 再計算するにあ a line of a table or the entire table, use the following commands: 次のコマンドを使ってください

*C-c **(`org-table-recalculate`)*
: Recalculate 再計算します the current row by first applying the stored column formulas from left to right, and all field/range formulas in the current row.

*C-u C-c ** or *C-u C-c C-c*
: Recompute the entire table, line by line. Any lines before the first hline are left alone, assuming that these are part of the table header.

*C-u C-u C-c ** or *C-u C-u C-c C-c (`org-table-iterate`)*
: Iterate the table by recomputing it until no further changes occur. This これが may be necessary 必用かもしれません if some computed fields use the value of other fields that are computed *later* 後のほうで in the calculation sequence.

M-x org-table-recalculate-buffer-tables
: Recompute all tables in the current buffer.

M-x org-table-iterate-buffer-tables
: Iterate all tables in the current buffer, in order to converge table-to-table dependencies.

3.5.10 Advanced features

If you want the recalculation of fields to happen automatically, 自動的に or if you want to be able to assign *names*[5] to fields and columns, you need to reserve 予約する必用がありますthe first column of the table for special marking characters.

C-# (`org-table-rotate-recalc-marks`)
: Rotate the calculation mark in first column through the states '`#`', '`*`', '`!`', '`$`'. When there is an active region, アクティブなリージョンがあるときには change 変更しますall marks 全てのマークを in the region. リージョンの中の

Here これは is an example of a table that collects exam results of students and makes use of these features:

```
|---+---------+--------+--------+--------+-------+------|
|   | Student | Prob 1 | Prob 2 | Prob 3 | Total | Note |
|---+---------+--------+--------+--------+-------+------|
| ! |         |     P1 |     P2 |     P3 |   Tot |      |
| # | Maximum |     10 |     15 |     25 |    50 | 10.0 |
| ^ |         |     m1 |     m2 |     m3 |    mt |      |
|---+---------+--------+--------+--------+-------+------|
| # | Peter   |     10 |      8 |     23 |    41 |  8.2 |
```

[5] Such names must start with an alphabetic character and use only alphanumeric/underscore characters.

```
| # | Sam     |      2 |      4 |      3 |     9 |  1.8 |
|---+---------+--------+--------+--------+-------+------|
|   | Average |        |        |        |  25.0 |      |
| ^ |         |        |        |        |    at |      |
| $ | max=50  |        |        |        |       |      |
|---+---------+--------+--------+--------+-------+------|
#+TBLFM: $6=vsum($P1..$P3)::$7=10*$Tot/$max;%.1f::$at=vmean(@-II..@-I);%.1f
```

Important: Please どうか note 注意してください that for these special tables, recalculating the table with *C-u C-c ** only affects rows that are marked '`#`' or '`*`', and fields that have a formula assigned to the field itself. The column formulas are not applied in rows with empty first field.

The marking characters have the following meaning:

'`!`' The fields フィールドは in this line この行の中の define 定義します names 名前を for the columns, so that you may refer 参照することができます to a column as '`$Tot`' として instead かわりに of '`$6`' の

'`^`' This row defines 定義します names for the fields *above* the row. With such a definition, そのような定義を使うと、any formula 全ての数式が in the table may use 使うことができますう'`$m1`' to refer to 参照するために the value '`10`'. Also, また if you assign 割り当てたら a formula 数式を to a names field, it is stored as '`$name = ...`'.

'`_`' Similar to '`^`' に似ていますが but defines 定義します names for the fields in the row *below*.

'`$`' Fields in this row can define *parameters* for formulas. For example, 例えば if a field in a '`$`' row contains 含んでいます'`max=50`', then formulas 数式は in this table can refer 参照することができます to the value 50 using '`$max`'. Parameters work exactly like constants, only that they can be defined on a per-table basis.

'`#`' Fields in this row are automatically 自動的に recalculated 再計算されます when pressing 押すときに`TAB` か`RET` か*S-TAB* を in this row. Also, また this row この行は is selected for a global recalculation with *C-u C-c **. Unmarked lines are left alone そのままにされます by this command.

'`*`' Selects this line for global recalculation with *C-u C-c **, but not for automatic recalculation. Use 使ってください this これを when automatic recalculation slows down editing too much.

'`/`' Do not export this line. Useful 役に立ちます for lines that contain the narrowing '`<N>`' markers or column group markers.

Finally, 最後に just to whet your appetite for what can be done with the fantastic Calc package, here is a table that computes the Taylor series of degree n at location x for a couple of functions.

```
|---+-------------+---+-----+--------------------------------------|
|   | Func        | n | x   | Result                               |
|---+-------------+---+-----+--------------------------------------|
| # | exp(x)      | 1 | x   | 1 + x                                |
| # | exp(x)      | 2 | x   | 1 + x + x^2 / 2                      |
```

```
| # | exp(x)      | 3 | x   | 1 + x + x^2 / 2 + x^3 / 6                |
| # | x^2+sqrt(x) | 2 | x=0 | x*(0.5 / 0) + x^2 (2 - 0.25 / 0) / 2 |
| # | x^2+sqrt(x) | 2 | x=1 | 2 + 2.5 x - 2.5 + 0.875 (x - 1)^2    |
| * | tan(x)      | 3 | x   | 0.0175 x + 1.77e-6 x^3               |
|---+-------------+---+-----+--------------------------------------|
#+TBLFM: $5=taylor($2,$4,$3);n3
```

3.6 Org Plot

Org Plot can produce graphs of information stored in Org tables, either graphically or in ASCII art.

Graphical plots using Gnuplot

Org Plot can produce 2D and 3D graphs of information stored in Org tables using Gnuplot (`http://www.gnuplot.info/`) and Gnuplot mode (`http://cars9.uchicago.edu/~ravel/software/gnuplot-mode.html`). To see this in action, ensure that you have both Gnuplot and Gnuplot mode installed on your system, then call *C-c " g* or *M-x org-plot/gnuplot* on the following table.

```
#+PLOT: title:"Citas" ind:1 deps:(3) type:2d with:histograms set:"yrange [0:]"
| Sede      | Max cites | H-index |
|-----------+-----------+---------|
| Chile     |    257.72 |   21.39 |
| Leeds     |    165.77 |   19.68 |
| Sao Paolo |     71.00 |   11.50 |
| Stockholm |    134.19 |   14.33 |
| Morelia   |    257.56 |   17.67 |
```

Notice that Org Plot is smart enough to apply the table's headers as labels. Further control over the labels, type, content, and appearance of plots can be exercised through the '`PLOT`' keyword preceding a table. See below 下を見てください。for a complete list 完全なリストについては of Org Plot のオプションの For more information and examples see the Org Plot tutorial (`https://orgmode.org/worg/org-tutorials/org-plot.html`) を見てください。

Plot options

'`set`' Specify any Gnuplot option to be set when graphing.

'`title`' Specify the title of the plot.

'`ind`' Specify which column of the table to use as the '`x`' axis.

'`deps`' Specify the columns to graph as a Lisp style list, surrounded by parentheses and separated by spaces for example 例えば'`dep:(3 4)`' to graph the third and fourth columns. Defaults デフォルトは to graphing グラフ化する all other columns 全ての列を aside from the '`ind`' column.

'`type`' Specify whether the plot is '`2d`', '`3d`', or '`grid`'.

'`with`' Specify a '`with`' option to be inserted for every column being plotted, e.g., 例えば'`lines`', '`points`', '`boxes`', '`impulses`'. デフォルトは'`lines`' です。

‘`file`’ If you want to plot to a file, specify 指定してください‘`"path/to/desired/output-file"`’ を

‘`labels`’ List of labels to be used for the ‘`deps`’. Defaults デフォルトは to the column headers if they exist.

‘`line`’ Specify an entire line to be inserted in the Gnuplot script.

‘`map`’ When plotting プロットするときには‘`3d`’ or ‘`grid`’ types, set this to ‘`t`’ to graph a flat mapping rather than a ‘`3d`’ slope.

‘`timefmt`’ Specify format of Org-mode timestamps as they will be parsed by Gnuplot. デフォルトは‘`%Y-%m-%d-%H:%M:%S`’ です。

‘`script`’ If you want total control, you can specify a script file---place the file name between double-quotes---which will be used to plot. Before plotting, every instance of ‘`$datafile`’ in the specified script will be replaced with the path to the generated data file. Note: even if you set this option, you may still それでも want to specify 指定しちかもいれまえん the plot type, as that can impact the content of the data file.

ASCII bar plots

While point is on a column, typing *C-c " a* か*M-x orgtbl-ascii-plot* create a new column containing 含んでいる an ASCII-art bars plot. ASCII アートのバープロットを The plot is implemented through a regular column formula. When the source column changes, the bar plot may be updated by refreshing the table, for example 例えば typing *C-u C-c **.

```
| Sede          | Max cites |              |
|---------------+-----------+--------------|
| Chile         |    257.72 | WWWWWWWWWWW  |
| Leeds         |    165.77 | WWWWWWWh     |
| Sao Paolo     |     71.00 | WWW;         |
| Stockholm     |    134.19 | WWWWWW:      |
| Morelia       |    257.56 | WWWWWWWWWWH  |
| Rochefourchat |      0.00 |              |
#+TBLFM: $3='(orgtbl-ascii-draw $2 0.0 257.72 12)
```

The formula is an Elisp call.

`orgtbl-ascii-draw` *value min max* **&optional** *width* [関数]

Draw an ASCII bar in a table.

VALUE is the value to plot.

MIN is the value displayed as an empty bar. *MAX* is the value filling all the *WIDTH*. Sources values outside this range are displayed as ‘`too small`’ or ‘`too large`’.

WIDTH is the number of characters of the bar plot. It defaults to ‘`12`’.

4 Hyperlinks

Like HTML, HTML と同じように、Org Org-mode は provides 提供しています support サポートを for links inside a file, external links to other files, Usenet articles, emails, and much more.

4.1 Link Format

Org recognizes plain URL-like links and activate them as clickable links. The general link format, 一般的なリンクのフォーマットは however, しかし looks like this: このように見えます:

```
[[LINK][DESCRIPTION]]
```

or alternatively

```
[[LINK]]
```

Once a link in the buffer is complete ---all brackets present —, Org Org-mode は changes 変更します the display 表示を so that '`DESCRIPTION`' が is displayed 表示され instead かわりに of '`[[LINK][DESCRIPTION]]`' の and '`LINK`' が is displayed instead of '`[[LINK]]`' のかわりに Links リンクは are highlighted ハイライトされます in the `org-link` というフェースで which, これは by default, デフォルトで is an underlined face. 下線つきのフェースです

You can directly edit the visible part of a link. This can be either the *LINK* part, if there is no description, or the *DESCRIPTION* part otherwise. そうでなければ To also edit the invisible *LINK* part, use 使ってください`C-c C-l` with point ポイントを on the link （節 4.5 [Handling Links], ページ 43, を見てください）。

If you place point at the beginning or just behind the end of the displayed text and press `BS`, you remove the---invisible---bracket at that location[1]. This makes the link incomplete and the internals are again displayed as plain text. Inserting the missing bracket hides the link internals again. To show the internal structure of all links, use the menu: Org → Hyperlinks → Literal links.

4.2 Internal Links

If the link does not look like a URL, it is considered to be internal in the current file. The most important case is a link like '`[[#my-custom-id]]`' which links to the entry with the '`CUSTOM_ID`' property '`my-custom-id`'. You are responsible yourself to make sure these custom IDs are unique in a file.

Links such as '`[[My Target]]`' や'`[[My Target][Find my target]]`' といった lead to a text search in the current file.

The link can be followed with `C-c C-o` when point is on the link, or with a mouse click （節 4.5 [Handling Links], ページ 43, を見てください）。Links to custom IDs point 指します to the corresponding headline. The preferred match for a text link is a *dedicated target*: the same string in double angular brackets, like '`<<My Target>>`'.

If no dedicated target exists, the link tries to match the exact name of an element within the buffer. Naming is done with the '`NAME`' keyword, which これは has to be put 入れられ

[1] More accurately, the precise behavior depends on how point arrived there---see 節 "Invisible Text" in `elisp`.

る必用があります in the line 行の中い before the element it refers to, as in the following example

```
#+NAME: My Target
| a  | table      |
|----+------------|
| of | four cells |
```

If none of the above succeeds, Org searches for a headline that is exactly the link text but may also include 含むこともできまう a TODO keyword and tags[2].

During export, internal links are used to mark objects and assign them a number. Marked objects are then referenced by links pointing to them. In particular, links リンクは without a description 説明のない appear as the number assigned to the marked object[3]. In the following excerpt from an Org buffer

```
1. one item
2. <<target>>another item
Here we refer to item [[target]].
```

The last sentence will appear as '`Here we refer to item 2`' when exported.

In non-Org files, the search looks for the words in the link text. In the above example the search would be for '`target`'.

Following a link pushes a mark onto Org's own mark ring. You can return to the previous position with *C-c &*. Using this command several times in direct succession goes back to positions recorded earlier.

4.3 Radio Targets

Org can automatically 自動的に turn 変えることができます any occurrences of certain target names in normal text into a link. So だから without explicitly creating 明示的に作成することなく a link, the text connects つながります to the target radioing its position. Radio targets are enclosed by triple angular brackets. For example, 例えば a target '`<<<My Target>>>`' というターゲットは causes each occurrence 各出現に of '`my target`' の in normal text to become activated as a link. リンクとして有効にします The Org file is scanned スキャナするえます automatically 自動的に for radio targets only when the file is first loaded into Emacs. To update the target list during editing, press *C-c C-c* with point on or at a target.

4.4 External Links

Org supports サポートしています links リンクを to files, websites, Usenet and email messages, BBDB database entries and links to both IRC conversations and their logs. External links are URL-like locators. They start with a short identifying string followed by a colon. There can be no space after the colon. The following list shows 示します examples 例を for each link type.

---

[2] To insert 挿入するには a link リンクを targeting a headline, 見出しをターゲットとしてる in-buffer completion バッファ内の補完を can be used. 使うことができます Just type a star followed by a few optional letters into the buffer and press *M-TAB*. All headlines in the current buffer are offered as completions.

[3] When targeting a '`NAME`' keyword, the '`CAPTION`' keyword is mandatory in order to get proper numbering （節 11.8 [Captions], ページ 144, を見てください）。

‘`http://www.astro.uva.nl/=dominik`’	on the web
‘`doi:10.1000/182`’	DOI for an electronic resource
‘`file:/home/dominik/images/jupiter.jpg`’	file, absolute path
‘`/home/dominik/images/jupiter.jpg`’	same as above
‘`file:papers/last.pdf`’	file, relative path
‘`./papers/last.pdf`’	same as above
‘`file:/ssh:me@some.where:papers/last.pdf`’	file, path on remote machine
‘`/ssh:me@some.where:papers/last.pdf`’	same as above
‘`file:sometextfile::NNN`’	file, jump to line number
‘`file:projects.org`’	another Org file
‘`file:projects.org::some words`’	text search in Org file[4]
‘`file:projects.org::*task title`’	heading search in Org file
‘`file+sys:/path/to/file`’	open via OS, like double-click
‘`file+emacs:/path/to/file`’	force opening by Emacs
‘`docview:papers/last.pdf::NNN`’	open in doc-view mode at page
‘`id:B7423F4D-2E8A-471B-8810-C40F074717E9`’	link to heading by ID
‘`news:comp.emacs`’	Usenet link
‘`mailto:adent@galaxy.net`’	mail link
‘`mhe:folder`’	MH-E folder link
‘`mhe:folder#id`’	MH-E message link
‘`rmail:folder`’	Rmail folder link
‘`rmail:folder#id`’	Rmail message link
‘`gnus:group`’	Gnus group link
‘`gnus:group#id`’	Gnus article link
‘`bbdb:R.*Stallman`’	BBDB link (with regexp)
‘`irc:/irc.com/#emacs/bob`’	IRC link
‘`info:org#External links`’	Info node link
‘`shell:ls *.org`’	shell command
‘`elisp:org-agenda`’	interactive Elisp command
‘`elisp:(find-file "Elisp.org")`’	Elisp form to evaluate

On top of these built-in link types, additional ones are available through the ‘`contrib/`’ directory（節 1.2 [Installation], ページ 2, を見てください）。For example, 例えば these links これらのリンクは to VM or Wanderlust messages are available when you load the corresponding libraries from the ‘`contrib/`’ directory:

‘`vm:folder`’	VM folder link
‘`vm:folder#id`’	VM message link
‘`vm://myself@some.where.org/folder#id`’	VM on remote machine
‘`vm-imap:account:folder`’	VM IMAP folder link
‘`vm-imap:account:folder#id`’	VM IMAP message link
‘`wl:folder`’	Wanderlust folder link
‘`wl:folder#id`’	Wanderlust message link

[4] The actual behavior of the search depends on the value of the variable 変数`org-link-search-must-match-exact-headline`. If its value is `nil`, then a fuzzy text search is done. If it is `t`, then only the exact headline is matched, ignoring spaces and statistic cookies. If the value is `query-to-create`, then an exact headline is searched; if it is not found, then the user is queried to create it.

For information on customizing Org to add new link types, see 節 A.3 [Adding Hyperlink Types], ページ 258, を見てください。

A link リンクは should be enclosed 囲まれるべきで in double brackets 大括弧の中に and may contain 含むことができます descriptive text 説明用のテキストを to be displayed 表示される instead of the URL のかわりに（節 4.1 [Link Format], ページ 40, を見てください）for example: 例えば

```
[[http://www.gnu.org/software/emacs/][GNU Emacs]]
```

If the description is a file name or URL that points to an image, HTML export（節 12.9 [HTML Export], ページ 159, を見てください）in lines the image as a clickable button. If there is no description at all and the link points to an image, that image is inlined into the exported HTML file.

Org also finds external links in the normal text and activates them as links. If spaces スペースが must be part 一部でなければならないのであれば、of the link リンクの(for example 例えば in '`bbdb:Richard Stallman`'), or if you need to remove 取り除く必用があれば ambiguities about the end of the link, enclose the link in square or angular brackets.

4.5 Handling Links

Org Org-mode は provides 提供しています methods 方法を to create a link in the correct syntax, to insert it into an Org file, and to follow the link.

The main function is `org-store-link`, called with *`M-x org-store-link`*. Because of its importance, we suggest to bind it to a widely available key（節 1.3 [Activation], ページ 3, を見てください）。It stores a link to the current location. The link is stored for later 後で insertion 挿入するために into an Org buffer ---下を見てください。The kind of link that is created 作成されます depends 依存します on the current buffer:

Org-mode buffers

> For Org files, if there is a '`<<target>>`' at point, the link points to the target. Otherwise そうでなければ it それは points 指します to the current headline, which これは is also the description 説明でもあります[5]。
>
> If the headline has a '`CUSTOM_ID`' property, store a link to this custom ID. In addition 加えてか

or alternatively, depending on 依存して the value 値に of `org-link-to-org-use-id` の create 作成し and/or use a globally unique グローバルにユニークな'`ID`' プロパティを for the link[6]. So だから using 使うと this command このコマンドを in Org buffers Org-mode のバッファの中で potentially 潜在的に creates 作成します two links: a human-readable link from the custom ID, and one that is globally unique グローバルにユニークで and works 動く even if the entry is moved from file to file. Later, 後で when inserting the link, you need to decide 決める必用があります which one to use. どちらを使うべきかを

---

[5] If the headline contains 含んでいます a timestamp, it is removed from the link, which これは results in 結果になります a wrong link ---you should avoid 避けるべきです putting 置くことを a timestamp タイムスタンプを in the headline. 見出しの中に

[6] The Org Id library must first be loaded, either through `org-customize`, by enabling `id` in `org-modules`, or by adding 追加することによって'`(require 'org-id)`' を in your Emacs init file.

Email/News clients: VM, Rmail, Wanderlust, MH-E, Gnus
: Pretty much all Emacs mail clients are supported. The link points to the current article, or, in some Gnus buffers, to the group. The description 説明は is constructed 構築されます according 従って to the 変数`org-email-link-description-format` に By default, デフォルトで it それは refers 指します to the addressee アドレスと and the subject. 題名を

Web browsers: W3, W3M and EWW
: Here the link is the current URL, with the page title as the description.

Contacts: BBDB
: Links created in a BBDB buffer point to the current entry.

Chat: IRC
: For IRC links, if the variable 変数`org-irc-link-to-logs` が`nil` でない値であれば、create 作成します a '`file`' style link to the relevant point in the logs for the current conversation. Otherwise そうでなければ store 保存します an '`irc`' style link to the user/channel/server under the point.

Other files
: For any other file, the link points to the file, with a search string（節 4.8 [Search Options], ページ 48, を見てください）pointing to the contents of the current line. If there is an active region, the selected words form the basis of the search string. You can write 書くことができます custom Lisp functions to select the search string and perform the search for particular file types（節 4.9 [Custom Searches], ページ 48, を見てください）。

  You can also define dedicated links to other files. 節 A.3 [Adding Hyperlink Types], ページ 258, を見てください。

Agenda view
: When point ポイントが is in an agenda view, アジェンダビューの中にあるときには、the created link 作成されるリンクは points 指します to the entry エントリーを referenced by the current line.

From an Org buffer, the following commands create, navigate or, more generally, act on links.

`C-c C-l` (`org-insert-link`)
: Insert 挿入します a link[7]. This これは prompts プロンプトを出します for a link リンクを求めて to be inserted 挿入するえる into the buffer. バッファの中に You can just type a link, using text for an internal link, or one of the link type prefixes mentioned in the examples above. The link is inserted into the buffer, along with ともに a descriptive text[8]. If some text was selected at this time, it becomes the default description.

[7] Note 注意してください that you do not have to use this command to insert a link. Links in Org are plain text, and you can type or paste them straight into the buffer. By using 使うことににによって this command, the links are automatically 自動的に enclosed 囲まれます in double brackets, and you will be asked for the optional descriptive text.

[8] After insertion of a stored link, the link will be removed from the list of stored links. To keep it それを in the list for later use, use a triple `C-u` prefix argument to `C-c C-l`, or configure 設定してください。the option `org-keep-stored-link-after-insertion` を

Inserting stored links

All links stored during the current session are part of the history for this prompt, このプロンプトに対する so you can access アクセスすることができます them with `UP` and `DOWN` (or *M-p*, *M-n*).

Completion support

Completion with `TAB` helps you to insert valid link prefixes like '`http`' や'`ftp`' のような including 含む the prefixes defined through link abbreviations（節 4.7 [Link Abbreviations], ページ 46, を見てください）。If you press `RET` を after inserting only the prefix, Org Org-mode は offers 提供します specific completion support 特定の補完のサポートを for some link types[9]. For example, 例えば if you type *f i l e RET* ---alternative access: *C-u C-c C-l*, see below ---Org Org-mode は offers 提供します file name completion, ファイル名の補完を and after *b b d b RET* you can complete contact names.

C-u C-c C-l

When *C-c C-l* が is called 呼ばれるときには with a *C-u* 前置引数といっしょに insert 挿入します a link to a file. You may use 使うことができまますう file name completion ファイル名の補完を to select the name of the file. The path to the file is inserted relative 相対的に to the directory of the current Org file, if the linked file is in the current directory or in a sub-directory of it, or if the path is written relative 相対的に to the current directory using 使って'`../`' を Otherwise そうでなければ an absolute path 絶対パスが is used, if possible 可能であれば with '`~/`' for your home directory. You can force an absolute path with two *C-u* prefixes.

C-c C-l (with point on existing link)

When point ポイントが is on an existing link, 既存のリンク上にあるときには、*C-c C-l* は allows 可能にします you to edit 編集することを the link リンク部分と and description parts 説明部分を of the link. リンクの

C-c C-o (`org-open-at-point`)

Open link at point. ポイントにあるリンクを開きます。This launches a web browser for URL (using `browse-url-at-point`), run VM/MH-E/Wanderlust/Rmail/Gnus/BBDB for the corresponding links, and execute the command in a shell link. When point ポイントが is on an internal link, 内部リンク上にあるときには、this command このコマンドは runs the corresponding search. 対応する検索を実行します。When point ポイントが is on the tags part of a headline, 見出しのタグ部分にあるときには、it それは creates 作成します the corresponding tags view（節 10.3.3 [Matching tags and properties], ページ 113, を見てください）。If point ポイントが is on a timestamp, タイムスタンプ上にあふるえば it それは compiles 編集します the agenda アジェンダを for that date. Furthermore, さらに、it visits text and remote files in '`file`' links with Emacs and select a suitable application for local non-text files. Classification of files is based on file extension only. オプ

[9] This works if a function has been defined in the `:complete` property of a link in `org-link-parameters`.

ションorg-file-apps を見てください。If you want to override 上書きしたいのであれば the default application デフォルトのアプリケーションを and visit the file with Emacs, use 使ってください a *C-u* 前置引数を If you want to avoid opening in Emacs, use 使ってください a *C-u C-u* 前置引数を

If point ポイントが is on a headline, 見出し上にあれば but not on a link, offer all links in the headline and entry text. If you want to setup セットアップしたいのであれば the frame configuration for following links, customize カスタマイズしてくださいorg-link-frame-setup を

RET When org-return-follows-link が is set, 設定されているときにはRET も also follows たどります the link at point.

mouse-2 or *mouse-1*
On links, *mouse-1* and *mouse-2* opens the link just as *C-c C-o* does.

mouse-3 Like *mouse-2* に似ていますが、but force 強制します file links to be opened with Emacs, and internal links to be displayed 表示されることを in another window[10].

C-c % (org-mark-ring-push)
Push the current position onto the Org mark ring, to be able to return easily. Commands following an internal link do this これを automatically. 自動的に行ないます

C-c & (org-mark-ring-goto)
Jump ジャンプしまづ back to a recorded position. A position is recorded by the commands following internal links, and by *C-c %*. Using this command several times in direct succession moves through a ring of previously recorded positions.

C-c C-x C-n (org-next-link)
C-c C-x C-p (org-previous-link)
Move 移動します forward/backward to the next link in the buffer. At the limit of the buffer, the search fails once, and then wraps around. The key bindings for this are really too long; you might want to bind this also to *M-n* and *M-p*.

```
(add-hook 'org-load-hook
          (lambda ()
            (define-key org-mode-map "\M-n" 'org-next-link)
            (define-key org-mode-map "\M-p" 'org-previous-link)))
```

4.6 Using Links Outside Org

You can insert 挿入して and follow たどることができます links リンクを that have Org syntax not only in Org, but in any Emacs buffer. For this, Org provides 提供しています two functions: 2 つの関数をorg-insert-link-global とorg-open-at-point-global です。

You might want to bind them to globally available keys. いくつかのアドバイスについては、節 1.3 [Activation], ページ 3, を見てください。

[10] See the variable 変数org-display-internal-link-with-indirect-buffer.

4.7 Link Abbreviations

Long URL can be cumbersome to type, タイプすることがやっかいなことがあり、and often many similar links 多くの似たようなリンクが are needed 必用ですす in a document. ドキュメントの中でこのためには、リンクの略語を使うことができます。略されたリンクはこのように見えます:

```
[[linkword:tag][description]]
```

where the tag is optional. The *linkword* must be a word, starting はじまり with a letter, 文字で followed by letters, numbers, '-', and '_'. Abbreviations are resolved according 従って to the information 情報に in the 変数`org-link-abbrev-alist` の中にある that relates the linkwords to replacement text. これは例です:

```
(setq org-link-abbrev-alist
      '(("bugzilla"  . "http://10.1.2.9/bugzilla/show_bug.cgi?id=")
        ("url-to-ja" . "http://translate.google.fr/translate?sl=en&tl=ja&u=%h")
        ("google"    . "http://www.google.com/search?q=")
        ("gmap"      . "http://maps.google.com/maps?q=%s")
        ("omap"      . "http://nominatim.openstreetmap.org/search?q=%s&polygon=1")
        ("ads"       . "http://adsabs.harvard.edu/cgi-bin/nph-abs_connect?author=%s&db
```

If the replacement text 置き換えのテキストが contains 含んでいたら the string 文字列'`%s`' を it is replaced with the tag. Using '`%h`' を使うと instead of '`%s`' のかわりに percent-encodes パーセントでエンコードします the tag タグを（上の例を見てください: where we need to encode the URL parameter）。Using '`%(my-function)`' passes the tag to a custom Lisp function, and replace it by the resulting string.

If the replacement text do not contain 含んでいなかったら any specifier, it is simply 単純に appended to the string in order to create the link.

Instead 文字列のんかわりに、of a string, you may also specify 指定することもできます a Lisp の関数を to create 作成するための the link. このリンクを Such a function will be called with the tag as the only argument.

With the above setting, 上の設定を使うと、you could link リンクすることができます to a specific bug with '`[[bugzilla:129]]`', search the web for '`OrgMode`' with '`[[google:OrgMode]]`', show the map location of the Free Software Foundation '`[[gmap:51 Franklin Street, Boston]]`' or of Carsten office '`[[omap:Science Park 904, Amsterdam, The Netherlands]]`' and find out what the Org author is doing besides Emacs hacking with '`[[ads:Dominik,C]]`'.

If you need 必用であれば special abbreviations 特別な略語が just for a single Org buffer, you can define them in the file with

```
#+LINK: bugzilla  http://10.1.2.9/bugzilla/show_bug.cgi?id=
#+LINK: google    http://www.google.com/search?q=%s
```

In-buffer completion バッファ内の補完を（節 15.1 [Completion], ページ 240, を見てください）can be used 使うことができます after '`[`' の後で to complete 補完するために link abbreviations. リンクの略語を You may also define 定義することもできます a Lisp function that implements special (e.g., 例えば completion) support for inserting such a link with `C-c C-l`. Such a function そのような関数は should not accept 受け入れるべきはなく any arguments, and should return 返すべきです the full link with a prefix. You can set 設定することができます the link completion function like this: このようにして

```
(org-link-set-parameter "type" :complete #'some-completion-function)
```

4.8 Search Options in File Links

File links can contain 含むことができます additional information to make Emacs jump to a particular location in the file when following a link. This can be a line number or a search option after a double colon[11]. For example, when the command `org-store-link` creates 作成するときには a link リンクを（節 4.5 [Handling Links], ページ 43, を見てください）to a file, ファイルへの it それは encodes エンコードします the words 単語を in the current line 現在の行の中にあう as a search string 検索文字列として that can be used 使うことができます to find 見つけるために this line この行を back later 後で when following たどるときに the link with *C-c C-o*.

Here is the syntax of the different ways to attach a search to a file link, together with explanations for each:

```
[[file:~/code/main.c::255]]
[[file:~/xx.org::My Target]]
[[file:~/xx.org::*My Target]]
[[file:~/xx.org::#my-custom-id]]
[[file:~/xx.org::/regexp/]]
```

‘`255`’ Jump ジャンプしまづ to line 255.

‘`My Target`’
Search for a link target ‘`<<My Target>>`’, or do a text search for ‘`my target`’, similar to the search in internal links, see 節 4.2 [Internal Links], ページ 40. In HTML export（節 12.9 [HTML Export], ページ 159, を見てください）such a file link becomes a HTML reference to the corresponding named anchor in the linked file.

‘`*My Target`’
In an Org file, restrict search to headlines.

‘`#my-custom-id`’
Link to a heading with a ‘`CUSTOM_ID`’ property

‘`/REGEXP/`’
Do a regular expression search for *REGEXP*. This これは uses 使います the Emacs のコマンド`occur` を to list リストするために all matches in a separate window. 別のウインドウの中で If the target file is in Org-mode, `org-occur` が is used 使われます to create 作成するために a sparse tree with the matches.

As a degenerate case, a file link with an empty file name can be used 使うことができます to search 検索するために the current file. 現在のファイルを For example, 例えば‘`[[file:::find me]]`’ は does 行ないます a search 検索を for ‘`find me`’ に対する in the current file, 現在のファイルの中の just as ‘`[[find me]]`’ とちょうど同じように、

[11] For backward compatibility, line numbers can also follow a single colon.

4.9 Custom Searches

The default mechanism デフォルトのメカニズムは for creating 作成するためと search strings 検索文字列を and for doing 行なうための the actual search 実際の検索を related to a file link may not work 動かないかもしれません correctly 正しく in all cases. 全ての場合には For example, 例えば BibTEX database files have あります many entries like `year="1993"` のような which would not result in good search strings, because the only unique identification for a BibTEX entry is the citation key. 引用キーなので

If you come across 出会ったら such a problem, そのような問題に you can write 書くことができます custom functions to set the right search string for a particular file type, and to do the search for the string in the file. Using `add-hook`, these functions need to be added to the hook variables フック変数`org-create-file-search-functions` と`org-execute-file-search-functions` に対してさらに情報については、これらの変数のドキュメント文字列を見てください。Org Org-mode は actually 実際に uses 使っていて、this mechanism このメカニズムを for BibTEX データベースファイルに対して and you can use 使うことができます the corresponding code 対応するコードを as an implementation example. 実装の例としてファイル‘`org-bibtex.el`’ を見てください。

5 TODO Items

Org-mode does not maintain TODO lists as separate documents[1]. Instead, かわりに TODO アイテムは are an integral part 統合されている一部です of the notes file, ノートファイルに because TODO アイテムが usually 通常は come up やってくるので while taking notes! ノートをとっている間に With Org-mode, Org-mode を使うと simply 単純に mark マークする any entry in a tree as being a TODO item. In this way, この方法で information is not duplicated, and the entire context from which the TODO アイテムが emerged 発生する is always 常に present. 存在します

Of course, this technique for managing TODO items scatters them throughout your notes file. Org-mode compensates for this by providing methods to give you an overview of all the things that you have to do.

5.1 Basic TODO Functionality

Any headline becomes a TODO item when it starts with the word '`TODO`', for example: 例えば

```
*** TODO Write letter to Sam Fortune
```

The most important commands to work with TODO entries are:

C-c C-t (`org-todo`)
: Rotate the TODO state of the current item among

```
,-> (unmarked) -> TODO -> DONE --.
'--------------------------------'
```

If TODO キーワードに have あれば fast access keys（節 5.2.4 [Fast access to TODO states], ページ 53, を見てください）、prompt プロンプトを出します for a TODO keyword TODO キーワードを求めて through the fast selection interface; this これは is the default behavior デフォルトのふるまいです when `org-use-fast-todo-selection` が is non-`nil`. `nil` でない値のときお

The same rotation can also be done 行なうこともできます"remotely" from the agenda buffer with the `t` command key（節 10.5 [Agenda Commands], ページ 121, を見てください）。

C-u C-c C-t
: When TODO キーワードに have ないときには no selection keys, 選択キーが select 選択します a specific keyword 特定のキーワードを using completion; otherwise そうでなければ force 強制します cycling through TODO states with no prompt. When `org-use-fast-todo-selection` が is set to `prefix` に設定されているときには、use 使います the fast selection interface.

S-RIGHT S-LEFT
: Select the following/preceding TODO state, similar to cycling. Useful 役に立ちます mostly 主に if more than two TODO states are possible 可能であれば（節 5.2 [TODO Extensions], ページ 51, を見てください）。See also 節 15.13.2 [Conflicts], ページ 252, も見てください。for a discussion the interaction with

[1] Of course, you can make a document that contains 含んでいます only long lists of TODO items, but this is not required.

やりとりの議論については、`shift-selection-mode` との変数`org-treat-S-cursor-todo-selection-as-state-change` も見てください。

C-c / t (`org-show-todo-tree`)
: View TODO items in a *sparse tree*（節 2.6 [Sparse Trees], ページ 12, を見てください）。Folds the entire buffer, but shows 表示します all TODO items 全ての TODO アイテムを---with not-DONE state ---and the headings hierarchy above them. With a prefix argument, 1 つの前置引数といっしょか、or by using 使うことによって*C-c / T* を、search 検索します for a specific TODO. 特定の TODO に対して You are prompted for the keyword, and you can also give a list of keywords like '`KWD1|KWD2|...`' to list entries that match any one of these keywords. With a numeric prefix argument N, show 表示します the tree for the Nth keyword in the variable 変数`org-todo-keywords` の中で With two prefix arguments, find all TODO states, both un-done and done.

M-x org-agenda t (`org-todo-list`)
: Show the global TODO list. Collects the TODO items (with not-DONE states) from all agenda files（章 10 [Agenda Views], ページ 108, を見てください）into a single buffer. The new buffer この新しいバッファは is in Org Agenda モードの中にあり、which これは provides 提供しています commands コマンドを to examine 確かめて and manipulate the TODO entries from the new buffer（節 10.5 [Agenda Commands], ページ 121, を見てください）。さらに情報については、節 10.3.2 [Global TODO list], ページ 112, を見てください。

S-M-RET (`org-insert-todo-heading`)
: Insert 挿入します a new TODO entry below the current one.

Changing 変更が a TODO state can also trigger トリガーをかけることもできます tag changes. タグの変更の詳細については、オプション`org-todo-state-tags-triggers` のドキュメント文字列を見てください。

5.2 Extended Use of TODO Keywords

By default, デフォルトで marked TODO entries have one of only two states: TODO and DONE です。Org-mode は allows 可能にしています you to classify 分類することを TODO アイテムを in more complex ways より複雑な方法で with *TODO keywords*（~org-todo-keywords~ の中に保存されます）。With special setup, the TODO keyword system can work 動くことができます differently in different files.

Note 注意してください that *tags/* are another way to classify headlines in general and TODO items in particular（章 6 [Tags], ページ 62, を見てください）。

5.2.1 TODO keywords as workflow states

You can use TODO キーワードを使うことができます。to indicate 示すために、different *sequential* states 異なる状態の順番 を in the process プロセスの中での of working on an item, あるアイテムに対して作業する for example 例えば

5.2.2 TODO keywords as types

The second possibility is to use TODO keywords to indicate different *types* of action items. For example, 例えば you might want to indicate that items are for "work" or "home". Or,

when you work with several people on a single project, you might want to assign action items directly to persons, by using 使うことによって their names as TODO keywords. This would be set up like this: このようにして

```
(setq org-todo-keywords '((type "Fred" "Sara" "Lucy" "|" "DONE")))
```

In this case, この場合には different keywords 異なるキーワードは do not indicate 示さず a sequence, シーケンスを but rather different types. 異なる種類を示します。So the normal work flow would be to assign a task to a person, and later 後で to mark it それを DONE. Org-mode は supports サポートしています this style このスタイルを by adapting the workings of the command *C-c C-t*[2]. When used 使われるときには、several times in succession, 複数回続けて it それは still それでも cycles through all names, in order to first select the right type for a task. But しかし when you return to the item after some time and execute *C-c C-t* を again, it will switch from any name directly to 'DONE'. Use 使ってください prefix arguments 前置引数か or completion 保存を to quickly 素早く select 選択するには a specific name. 特定の名前を You can also review the items of a specific TODO type in a sparse tree by using 使うことによって a numeric prefix 数値の前置引数を to *C-c / t* に対して For example, 例えば to see all things Lucy has to do, you would use *C-3 C-c / t*. To collect 集めるには Lucy のアイテムを from all agenda files into a single buffer, 1 つのバッファの中へ you would use 使います the numeric prefix argument as well when creating the global TODO list: *C-3 M-x org-agenda t* です。

5.2.3 Multiple keyword sets in one file

Sometimes ときどき you may want to use 使いたいことがありまう different sets 異なる集合を of TODO キーワードの in parallel. 平行して For example, 例えば you may want to have the basic TODO/DONE, but also a workflow for bug fixing, and a separate state indicating that an item has been canceled ---so it is not DONE, but also does not require action. Your setup would then look like this: このように見めます

```
(setq org-todo-keywords
      '((sequence "TODO" "|" "DONE")
        (sequence "REPORT" "BUG" "KNOWNCAUSE" "|" "FIXED")
        (sequence "|" "CANCELED")))
```

The keywords should all be different, 全て異なるべきですう this helps Org-mode keep track of which subsequence should be used 使われるべきか for a given entry. In this setup, *C-c C-t* は only operates within a sub-sequence, so it switches from 'DONE' to (nothing) to 'TODO', and from 'FIXED' to (nothing) to 'REPORT'. Therefore you need a mechanism to initially select the correct sequence. In addition 加えて to typing タイプすることに a keyword キーワードを or using completion（節 15.1 [Completion], ページ 240, を見てください）you may also apply 適用することもできます the following commands: 次の関数を

C-u C-u C-c C-t
C-S-RIGHT
C-S-LEFT These keys jump from one TODO sub-sequence to the next. In the above example, *C-u C-u C-c C-t* or *C-S-RIGHT* would jump from 'TODO' or 'DONE' to 'REPORT', and any of the words in the second row to 'CANCELED'. Note 注意してください that the *C-S-* key binding conflict 衝突します with `shift-selection-mode` と（節 15.13.2 [Conflicts], ページ 252, を見てください）。

[2] This is also true for the `t` command in the agenda buffer.

S-RIGHT
S-LEFT *S-LEFT* and *S-RIGHT* walk through *all* keywords from all sub-sequences, so for example 例えば*S-RIGHT* would switch from 'DONE' to 'REPORT' in the example above. For a discussion of the interaction with `shift-selection-mode`, see [BROKEN LINK: *Packages that conflict with Org mode] を見てください。

5.2.4 Fast access to TODO states

If you would like to quickly 素早く change 変更したいのであればん an entry エントリーを to an arbitrary TODO state 任意の TODO の状態へ instead かわりに of cycling 切り替える through the states, you can set up セットアップすることができます keys for single-letter access to the states. This is done 行なわれます by adding 追加することによって the selection character 選択用文字を after each keyword, in parentheses[3]. For example:

```
(setq org-todo-keywords
      '((sequence "TODO(t)" "|" "DONE(d)")
        (sequence "REPORT(r)" "BUG(b)" "KNOWNCAUSE(k)" "|" "FIXED(f)")
        (sequence "|" "CANCELED(c)")))
```

If you then press *C-c C-t* を押して followed 続けたら by the selection key, 選択キーを the entry このエントリーは is switched 切り替えられます to this state. この状態へ`SPC` を can be used 使うことができます to remove 取り除くために any TODO キーワードを from an entry エントリーから

5.2.5 Setting up keywords for individual files

It can be very useful とても役に立つことがります to use 使うということが different aspects of the TODO mechanism in different files. For file-local settings, you need to add 追加する必用があります special lines to the file which set the keywords and interpretation for that file only. For example, 例えば to set one of the two examples discussed above, you need one of the following lines, starting in column zero anywhere in the file:

```
#+TODO: TODO FEEDBACK VERIFY | DONE CANCELED
```

You may also write 書くこともできま'`#+SEQ_TODO`' と to be explicit about the interpretation, but it means 意味します the same as '`#+TODO`', or

```
#+TYP_TODO: Fred Sara Lucy Mike | DONE
```

A setup for using several sets in parallel would be:

```
#+TODO: TODO | DONE
#+TODO: REPORT BUG KNOWNCAUSE | FIXED
#+TODO: | CANCELED
```

To make sure 確実にするには you are using 使っていることを the correct keyword, 正しいキーワードを type タイプして'`#+`' を into the buffer バッファの中に and then 次ん use 使うということです*M-TAB* を to complete it それを補完するために（節 15.1 [Completion], ページ 240, を見てください）。

Remember that the keywords after the vertical bar ---or the last keyword if no bar is there ---must always 常に mean 意味しなければならない that the item そのアイテムが is DONE であるこていお although you may use 使うことができますが a different word.

[3] All characters are allowed except '`@`', '`^`' and '`!`', which これらには have あります a special meaning here.

After changing 変更した後 one of these lines, use `C-c C-c` を with point still in the line to make the changes known to Org-mode[4].

5.2.6 Faces for TODO keywords

Org-mode highlights ハイライトします TODO keywords with special faces: `org-todo` for keywords indicating 示している that an item still has to be acted upon, and `org-done` for keywords indicating that an item is finished. If you are using more than two different states, you might want to use 使うとよいかもしれません special faces for some of them. This can be done using 使って the variable 変数`org-todo-keyword-faces` を例です:

```
(setq org-todo-keyword-faces
      '(("TODO" . org-warning) ("STARTED" . "yellow")
        ("CANCELED" . (:foreground "blue" :weight bold))))
```

While using a list with face properties as shown for '`CANCELED`' *should* work, 動くはずですが this これは does not always 常に seem 思えません to be the case. 適切であるとは If necessary, 必用であれば define a special face and use that. A string is interpreted as a color. The variable 変数`org-faces-easy-properties` が determines 決定しまますう if that color is interpreted as a foreground or a background color.

5.2.7 TODO dependencies

The structure of Org files---hierarchy and lists---makes it easy to define TODO dependencies. Usually, 通常は a parent TODO task should not be marked マークされるべきではりません DONE と until all TODO subtasks, or children tasks, are marked as DONE. Sometimes ときどき there is a logical sequence to (sub)tasks, so that one subtask cannot be acted upon before all siblings above it have been marked DONE. If you customize カスタマイズしたら the 変数`org-enforce-todo-dependencies` を Org blocks entries from changing state to DONE while they have TODO children that are not DONE. Furthermore, さらに、if an entry has a property '`ORDERED`', each of its TODO children is blocked until all earlier siblings are marked DONE. Here is an example:

```
* TODO Blocked until (two) is done
** DONE one
** TODO two

* Parent
:PROPERTIES:
:ORDERED:  t
:END:
** TODO a
** TODO b, needs to wait for (a)
** TODO c, needs to wait for (a) and (b)
```

You can ensure an entry is never blocked by using 使うことによって the '`NOBLOCKING`' プロパティを（章 7 [Properties and Columns], ページ 66, を見てください）。

```
* This entry is never blocked
:PROPERTIES:
```

[4] Org-mode parses these lines only when Org-mode is activated after visiting a file. `C-c C-c` with point in a line starting with '`#+`' is simply 単純に restarting 再スタートします Org-mode for the current buffer.

```
:NOBLOCKING: t
:END:
```

C-c C-x o (`org-toggle-ordered-property`)
: Toggle トグルします the '`ORDERED`' property of the current entry. A property is used for this behavior because this should be local to the current entry, not inherited from entries above like a tag（章 6 [Tags], ページ 62, を見てください）。However, しかし if you would like to *track* the value of this property with a tag for better visibility, customize カスタマイズしてください the 変数`org-track-ordered-property-with-tag` を

C-u C-u C-u C-c C-t
: Change TODO state, regardless of any state blocking.

If you set the variable 変数`org-agenda-dim-blocked-tasks` を TODO entries that cannot be marked DONE と because of unmarked children are shown in a dimmed font or even made invisible in agenda views（章 10 [Agenda Views], ページ 108, を見てください）。

You can also block changes of TODO states by using 使うことによって checkboxes チェックボックスを（節 5.6 [Checkboxes], ページ 59, を見てください）。If you set the variable 変数`org-enforce-todo-checkbox-dependencies` を an entry that has unchecked checkboxes is blocked from switching to DONE.

If you need more complex dependency structures, for example 例えば dependencies between entries in different trees or files, check out the contributed module '`org-depend.el`'.

5.3 Progress Logging

Org-mode は can automatically 自動的に record 記録することができます a timestamp タイムスタンプと and optionally オプションで a note ノートを when you mark a TODO item as DONE, DONE として or even each time you change the state of a TODO item. This system is highly configurable, settings can be on a per-keyword basis and can be localized to a file or even a subtree. For information on how to clock working time for a task, see 節 8.4 [Clocking Work Time], ページ 82.

5.3.1 Closing items

The most basic logging is to keep track of *when* a certain TODO item was marked DONE. This can be achieved with[5]

```
(setq org-log-done 'time)
```

Then each time you turn an entry from a TODO (not-done) state into any of the DONE states, a line '`CLOSED: [timestamp]`' is inserted just after the headline. If you turn the entry back into a TODO item through further state cycling, that line is removed again. If you turn the entry back to a non-TODO state (by pressing *C-c C-t SPC* を押すことによって for example), 例えば that line is also removed, unless you set 設定しないかぎり`org-closed-keep-when-no-todo` を to non-`nil`. `nil` でない値に If you want to record a note along with ともに the timestamp, タイムスタンプと use[6]

```
(setq org-log-done 'note)
```

[5] The corresponding in-buffer setting is: '`#+STARTUP: logdone`'.

[6] The corresponding in-buffer setting is: '`#+STARTUP: lognotedone`'.

You are then be prompted for a note, and that note is stored below the entry with a '`Closing Note`' heading.

5.3.2 Tracking TODO state changes

When TODO キーワードが are used as workflow states ワークフローの状態として使われるときには、(節 5.2.1 [*Workflow states], ページ 51, を見てください)。you might want to keep track of when いつ a state change occurred and maybe take a note about this change. You can either record just a timestamp, or a time-stamped note. These records are inserted after the headline as an itemized list, newest first[7]. When taking a lot of notes, たくさんのノートをとるときには you might want to get the notes ノートを out of the way into a drawer (節 2.8 [Drawers], ページ 16, を見てください)。Customize カスタマイズしてください the variable 変数`org-log-into-drawer` を to get this behavior このふるまいを得るには---the recommended drawer for this is called '`LOGBOOK`'

5.3.3 Tracking your habits

Org Org-mode には has あります the ability 機能が to track 追うための the consistency 一貫性を of a special category of TODO, TODO の特別なカテゴリーの called "habits" と呼ばれる To use 使うには、habits, 習慣を you have to enable 有効にする必用がありますthe `habits` モジュールを module by customizing カスタマイズすることによって、the 変数`org-modules` を

A habit has the following properties:

1. The habit is a TODO item, with a TODO keyword representing an open state.
2. The property '`STYLE`' is set to the value '`habit`' (章 7 [Properties and Columns], ページ 66, を見てください)。
3. The TODO has a scheduled date, usually 通常は with a '`.+`' style repeat interval. A '`++`' style may be appropriate 適切かもしれません for habits 週間に対しては with time constraints, e.g., 例えば must be done 行なわれなければならない on weekends, or a '`+`' style for an unusual habit that can have a backlog, e.g., 例えば weekly reports.
4. The TODO may also have minimum and maximum ranges specified by using 使うことによって the syntax '`.+2d/3d`' という文法を which これは says 言っています that you want to do the task そのタスクを at least every three days, but at most every two days.
5. State logging for the DONE state is enabled (節 5.3.2 [Tracking TODO state changes], ページ 56, を見てください)、in order for historical data to be represented in the consistency graph. If it is not enabled それが有効でなくても it is not an error, エラーではありませんが but the consistency graphs 一貫性のグラフには are largely meaningless. 大きく意味がなくなりませす

To give 与えるために you an idea of what the above rules 上のルールが look like in action, 動く様子を here's an actual habit with some history:

```
** TODO Shave
   SCHEDULED: <2009-10-17 Sat .+2d/4d>
   :PROPERTIES:
   :STYLE:    habit
```

[7] See the variable 変数`org-log-states-order-reversed`.

```
:LAST_REPEAT: [2009-10-19 Mon 00:36]
:END:
- State "DONE"       from "TODO"       [2009-10-15 Thu]
- State "DONE"       from "TODO"       [2009-10-12 Mon]
- State "DONE"       from "TODO"       [2009-10-10 Sat]
- State "DONE"       from "TODO"       [2009-10-04 Sun]
- State "DONE"       from "TODO"       [2009-10-02 Fri]
- State "DONE"       from "TODO"       [2009-09-29 Tue]
- State "DONE"       from "TODO"       [2009-09-25 Fri]
- State "DONE"       from "TODO"       [2009-09-19 Sat]
- State "DONE"       from "TODO"       [2009-09-16 Wed]
- State "DONE"       from "TODO"       [2009-09-12 Sat]
```

What this habit says is: I want to shave at most every 2 days---given by the '`SCHEDULED`' date and repeat interval---and at least every 4 days. If today is the 15th, then the habit first appears in the agenda（章 10 [Agenda Views], ページ 108, を見てください）on Oct 17, after the minimum of 2 days has elapsed, and will appear overdue on Oct 19, after four days have elapsed.

What's really useful 本当に役に立つことは about habits is that they are displayed along with a consistency graph, to show how consistent you've been at getting that task done in the past. This graph shows 表示します every day 全ての日を that the task このタスクが was done DONE として over the past three weeks, 過去 3 週間で with colors for each day. The colors used are:

Blue If the task was not to be done yet on that day.

Green If the task could have been done on that day.

Yellow If the task was going to be overdue the next day.

Red If the task was overdue on that day.

In addition 加えて to coloring each day, the day is also marked with an asterisk if the task was actually done that day, and an exclamation mark to show where the current day falls in the graph.

There are several configuration variables 複数の設定用変数があります that can be used 使うことができます to change 変更するために the way 方法を habits 収集が are displayed in the agenda. アジェンダの中で表示される

`org-habit-graph-column`
: The buffer column at which the consistency graph should be drawn. This overwrites any text in that column, so it is a good idea to keep your habits' titles brief and to the point.

`org-habit-preceding-days`
: The amount of history, in days before today, to appear in consistency graphs.

`org-habit-following-days`
: The number of days after today that appear in consistency graphs.

`org-habit-show-habits-only-for-today`
: If non-`nil`, `nil` でない値であれば only show habits in today's agenda view. デフォルト値は`t` です。

Lastly, pressing 押すと*K* を in the agenda buffer causes habits 習慣に to temporarily be disabled and do not appear at all. Press *K* again もう一度 to bring them back. They are also subject to tag filtering, if you have habits which should only be done in certain contexts, for example.

5.4 Priorities

If you use Org-mode extensively, you may end up with enough TODO items that it starts to make sense 意味が出はじめあますう to prioritize them. Prioritizing can be done 行なうことができます by placing 置くことによっ a *priority cookie* into the headline of a TODO item, like this このようにして

```
*** TODO [#A] Write letter to Sam Fortune
```

By default, デフォルトで Org-mode は supports サポートしています three priorities: '`A`', '`B`', and '`C`'. '`A`' is the highest priority. An entry without a cookie クッキーのない is treated as equivalent 同じであると if it had priority '`B`'. Priorities make a difference 違いが出ます only for sorting in the agenda（節 10.3.1 [Weekly/daily agenda], ページ 110, を見てください）; outside the agenda, they have no inherent meaning to Org-mode. The cookies are displayed 表示されます with the face defined by the variable 変数`org-priority-faces` によって which can be customized. カスタマイズすることができます

Priorities can be attached to any outline node; they do not need to be TODO items.

C-c , (`org-priority`)
: Set the priority of the current headline. The command このコマンドは prompts プロンプトを出します for a priority character '`A`', '`B`' or '`C`'. When you press `SPC` を押すときには、instead, かわりに the priority cookie, 優先順位のクッキーは if one is set, is removed from the headline. その見出しから取り除かれます The priorities can also be changed "remotely" from the agenda buffer with the `,` command（節 10.5 [Agenda Commands], ページ 121, を見てください）。

S-UP (`org-priority-up`)
S-DOWN (`org-priority-down`)
: Increase/decrease the priority 優先順位を of the current headline[8]. Note 注意してください that these keys are also used to modify timestamps（節 8.2 [Creating Timestamps], ページ 76, を見てください）。See also 節 15.13.2 [Conflicts], ページ 252, も見てください。for a discussion 議論については、of the interaction with `shift-selection-mode` とのやりとりの

You can change 変更することができます the range of allowed priorities 許可されている優先順位の範囲を by setting 設定することによって the 変数`org-highest-priority` と`org-lowest-priority` と`org-default-priority` を For an individual buffer, 個々のバッファに対しては、you may set 設定することができます these values これらの値を(highest, lowest, default) like this このようにして(please どうか make sure 確実いしてください that the highest priority is earlier in the alphabet than the lowest priority):

```
#+PRIORITIES: A C B
```

[8] See also the option `org-priority-start-cycle-with-default`.

5.5 Breaking Down Tasks into Subtasks

It is often advisable to break down large tasks into smaller, manageable subtasks. You can do this これを by creating an outline tree below a TODO item, with detailed subtasks on the tree[9]. To keep 保つたえんい an overview 概要を of the fraction of subtasks that have already been marked DONE, insert either '`[/]`' or '`[%]`' anywhere in the headline. These cookies are updated each time the TODO status of a child changes, or when pressing *C-c C-c* on the cookie. 例です:

```
* Organize Party [33%]
** TODO Call people [1/2]
*** TODO Peter
*** DONE Sarah
** TODO Buy food
** DONE Talk to neighbor
```

If a heading has both checkboxes and TODO children below it, the meaning of the statistics cookie become ambiguous. あいまいになります Set the property '`COOKIE_DATA`' to either '`checkbox`' or '`todo`' to resolve this issue. この問題を解決するには、

If you would like to have the statistics cookie count 数えさせたいのであれば any TODO entries in the subtree (not just direct children), configure 設定してください the 変数`org-hierarchical-todo-statistics` を To do this これを行うには for a single subtree, 1 つのサブツリーに対して include 含めてください the word '`recursive`' という単語を into the value 値の中に of the '`COOKIE_DATA`' プロパティの

```
* Parent capturing statistics [2/20]
  :PROPERTIES:
  :COOKIE_DATA: todo recursive
  :END:
```

If you would like a TODO エクスポートが to automatically change 自動的に変化することを望むのであれば、to DONE へ when all children 全ての子が are done, 終了したときに you can use the following setup: 次のセットアップを使うことができます:

```
(defun org-summary-todo (n-done n-not-done)
  "Switch entry to DONE when all subentries are done, to TODO otherwise."
  (let (org-log-done org-log-states)   ; turn off logging
    (org-todo (if (= n-not-done 0) "DONE" "TODO"))))

(add-hook 'org-after-todo-statistics-hook 'org-summary-todo)
```

Another possibility is the use of checkboxes to identify (a hierarchy of) a large number of subtasks（節 5.6 [Checkboxes], ページ 59, を見てください）。

5.6 Checkboxes

Every item in a plain list[10]（節 2.7 [Plain Lists], ページ 13, を見てください）can be made into a checkbox by starting it with the string '`[ ]`' という文字列ではじめることによって This feature is この機能は similar to TODO items TODO アイテムに似ていますが（章 5

[9] To keep subtasks out of the global TODO list, see the option `org-agenda-todo-list-sublevels`.

[10] With the exception of description lists. But しかし you can allow it by modifying `org-list-automatic-rules` accordingly.

[TODO Items], ページ 50, を見てください）、but is more lightweight. より軽量です Checkboxes are not included into the global TODO list, so they are often しばしば great to split a task into a number of simple steps. Or you can use them in a shopping list. To toggle トグルするには a checkbox, チェックボックスを use `C-c C-c` を使うか or use 使ってください the mouse マウスを(thanks to Piotr Zielinski's '`org-mouse.el`').

Here is an example of a checkbox list.

```
* TODO Organize party [2/4]
  - [-] call people [1/3]
    - [ ] Peter
    - [X] Sarah
    - [ ] Sam
  - [X] order food
  - [ ] think about what music to play
  - [X] talk to the neighbors
```

Checkboxes work hierarchically, so if a checkbox item has children that are checkboxes, toggling one of the children checkboxes makes the parent checkbox reflect if none, some, or all of the children are checked.

The '`[2/4]`' and '`[1/3]`' in the first and second line are cookies indicating how many checkboxes present in this entry have been checked off, and the total number of checkboxes present. This can give you an idea on how many checkboxes remain, even without opening 開くことさえなく a folded entry. The cookies can be placed into a headline or into (the first line of) a plain list item. Each cookie covers checkboxes of direct children structurally below the headline/item on which the cookie appears[11]. You have to insert the cookie yourself by typing either '`[/]`' or '`[%]`'. With '`[/]`' you get an '`n out of m`' result, as in the examples above. With '`[%]`' you get information about the percentage of checkboxes checked (in the above example, this would be '`[50%]`' and '`[33%]`', respectively). In a headline, a cookie can count either checkboxes below the heading or TODO states of children, and it displays 表示します whatever was changed last. Set the property '`COOKIE_DATA`' to either '`checkbox`' or '`todo`' to resolve this issue.

If the current outline node has an '`ORDERED`' property, checkboxes must be checked off in sequence, and an error is thrown if you try to check off a box while there are unchecked boxes above it.

The following commands work with checkboxes:

`C-c C-c` (`org-toggle-checkbox`)
: Toggle トグルします checkbox status or---with prefix argument---checkbox presence at point. With a single prefix argument, add an empty checkbox or remove the current one[12]. With a double prefix argument, set it to '`[-]`', which これは is considered みなされます to be an intermediate state. 中間状態であると

[11] Set the variable 変数`org-hierarchical-checkbox-statistics` if you want such cookies to count all checkboxes below the cookie, not just those belonging to direct children.

[12] `C-u C-c C-c` on the *first* item of a list with no checkbox adds checkboxes to the rest of the list.

C-c C-x C-b (`org-toggle-checkbox`)
: Toggle トグルします checkbox status or---with prefix argument---checkbox presence at point. With double prefix argument, set it to '`[-]`' へ which これは is considered to be an intermediate state. 中間状態であるとみなされます

  - If there is an active region, toggle the first checkbox in the region and set all remaining boxes to the same status as the first. With a prefix argument, 1 つの前置引数といっしょだと、add or remove the checkbox for all items in the region.
  - If point ポイントが is in a headline, 見出しの中にあれば toggle トグルします checkboxes in the region between this headline and the next---so *not* the entire subtree.
  - If there is no active region, just toggle the checkbox at point.

M-S-RET (`org-insert-todo-heading`)
: Insert 挿入します a new item with a checkbox. This works only if point is already in a plain list item（節 2.7 [Plain Lists], ページ 13, を見てください）。

C-c C-x o (`org-toggle-ordered-property`)
: Toggle トグルします the '`ORDERED`' property of the entry, to toggle if checkboxes must be checked off オフにされなければならないのであふるえば in sequence. 順番に A property is used for this behavior because this should be local to the current entry, not inherited like a tag. However, しかし if you would like to *track* the value of this property with a tag for better visibility, customize カスタマイズしてくださいorg-track-ordered-property-with-tag を

C-c # (`org-update-statistics-cookies`)
: Update the statistics cookie in the current outline entry. When called 呼ばれるときには with a *C-u* 前置引数といっしょに update 更新します the entire file. ファイル全体を Checkbox statistic cookies are updated automatically 自動的に更新されます if you toggle checkboxes with *C-c C-c* and make new ones with *M-S-RET*. TODO statistics cookies update when changing TODO states. If you delete boxes/entries or add/change them by hand, use this command to get things back into sync.

6 Tags

An excellent way to implement labels and contexts for cross-correlating information is to assign *tags* to headlines. Org mode has extensive support for tags.

Every headline can contain 含むことができまますう a list of tags; they occur at the end of the headline. Tags are normal words containing 含んでいる letters, 文字と numbers, 数字、‘`_=、and =@`’ を Tags must be preceded and followed by a single colon, e.g., 例えば‘`:work:`’. Several tags can be specified, 指定することができます as in ‘`:work:urgent:`’ といったように、Tags タグは by default デフォルトで are in bold face with the same color as the headline. You may specify 指定することができあますう special faces 特別なフェースを for specific tags using 変数`org-tag-faces` を使って in much the same way ほとんど同じ方法で as you can for TODO keywords（節 5.2.6 [Faces for TODO keywords], ページ 54, を見てください）。

6.1 Tag Inheritance

Tags make use of the hierarchical structure of outline trees. If a heading ある見出しに has a certain tag, all subheadings inherit the tag as well. For example, 例えば in the list

```
* Meeting with the French group      :work:
** Summary by Frank                  :boss:notes:
*** TODO Prepare slides for him      :action:
```

the final heading has the tags ‘`work`’, ‘`boss`’, ‘`notes`’, and ‘`action`’ even though the final heading is not explicitly marked with those tags. You can also set tags that all entries in a file should inherit just as if these tags were defined in a hypothetical level zero that surrounds the entire file. Use 使ってください a line like this このようなリンクを

6.2 Setting Tags

Tags タグを can simply 単純に be typed タイプすることげんえきます into the buffer at the end 終わりに of a headline. 見出しの After a colon, コロンの後で*`M-TAB`* は offers 提供します completion 補完を on tags. There is also a special command for inserting tags:

`C-c C-q` (`org-set-tags-command`)
: Enter new tags for the current headline. Org-mode either offers 提供します completion 補完か or a special single-key interface for setting tags, タグを設定するために see below. 下を見てください。After pressing `RET` を押した後で、the tags are inserted and aligned to `org-tags-column`. When called with a *`C-u`* 前置引数といっしょに呼ばれるときには、all tags 全てのタグが in the current buffer are aligned to that column, just to make things look nice. Tags タグは are automatically 自動的に realigned 再整列されます after promotion, demotion, and TODO state changes（節 5.1 [TODO Basics], ページ 50, を見てください）。

`C-c C-c` (`org-set-tags-command`)
: When point ポイントが見出しの上にあるときには、is in a headline, this これは does the same as *`C-c C-q`* と同じことを行ないます。

Org Org-mode は supports サポートしています tag insertion based on a *list of tags*. By default デフォルトで this list このリストは is constructed dynamically, containing 含んでいる all tags 全てのタグを currently used 現在使われている in the buffer バッファの中で

6.3 Tag Hierarchy

Tags can be defined in hierarchies. A tag can be defined as a *group tag* for a set of other tags. The group tag can be seen as the "broader term" for its set of tags. Defining multiple group tags and nesting them creates 作成します a tag hierarchy. タグの階層構造を

One use-case 1 つのユースケースは is to create 作成するということです a taxonomy of terms (tags) that can be used 使うことができます to classify 分類するために nodes ノードを in a document ドキュメントや or set of documents. 複数のドキュメントの中にある

When you search 検索するときには for a group tag, グループタグに対して it それは return 返します matches for all members in the group and its subgroups. In an agenda view, filtering by a group tag displays 表示します or hide 隠します headlines tagged with at least one of the members of the group or any of its subgroups. This makes tag searches and filters even more flexible.

You can set 設定することができます group tags グループタグを by using 使うことによって brackets 大括弧を and inserting 挿入することによって a colon between the group tag and its related tags---beware that all whitespaces are mandatory so that Org can parse this line correctly:

```
#+TAGS: [ GTD : Control Persp ]
```

In this example, ‘GTD’ is the group tag and it is related to two other tags: ‘Control’, ‘Persp’. Defining ‘Control’ and ‘Persp’ as group tags creates 作成します a hierarchy of tags: タグの階層構造を

```
#+TAGS: [ Control : Context Task ]
#+TAGS: [ Persp : Vision Goal AOF Project ]
```

That can conceptually be seen as a hierarchy of tags:

- ‘GTD’
  - ‘Persp’
    - ‘Vision’
    - ‘Goal’
    - ‘AOF’
    - ‘Project’
  - ‘Control’
    - ‘Context’
    - ‘Task’

You can use 使うことができます the キーワード:startgrouptag と:grouptags と:endgrouptag を directly 直接 when setting 設定するときにorg-tag-alist を directly: 直接

```
(setq org-tag-alist '((:startgrouptag)
                      ("GTD")
                      (:grouptags)
```

```
("Control")
("Persp")
(:endgrouptag)
(:startgrouptag)
("Control")
(:grouptags)
("Context")
("Task")
(:endgrouptag)))
```

The tags in a group can be mutually exclusive if using the same group syntax as is used for grouping mutually exclusive tags together; using curly brackets.

```
#+TAGS: { Context : @Home @Work @Call }
```

When setting `org-tag-alist` を設定するときには、you can use `:startgroup` と`:endgroup` を使うことができます。instead of `:startgrouptag` と`:endgrouptag` のかわりに to make the tags タグを mutually exclusive. 互いに排他的にするために、

Furthermore, さらに、the members of a group tag can also be regular expressions, creating the possibility of a more dynamic and rule-based tag structure. The regular expressions in the group must be specified 指定されなければなりません within curly brackets. Here is an expanded example:

```
#+TAGS: [ Vision : {V@.+} ]
#+TAGS: [ Goal : {G@.+} ]
#+TAGS: [ AOF : {AOF@.+} ]
#+TAGS: [ Project : {P@.+} ]
```

Searching for the tag '`Project`' now lists all tags also including 含む regular expression matches for '`P@.+`', and similarly for tag searches on '`Vision`', '`Goal`' and '`AOF`'. For example, 例えば this would work well for a project tagged with a common project-identifier, e.g., 例えば'`P@2014_OrgTags`'.

If you want to ignore group tags temporarily, toggle group tags support with `org-toggle-tags-groups`, bound to *C-c C-x q*. If you want to disable tag groups completely, set `org-group-tags` to `nil`.

6.4 Tag Searches

Once a system of tags has been set up, タグのシステムがセットアップされたら、it それを使うことができます。can be used to collect related information 関係する情報を集めるために、into special lists. 特別なリストの中に

C-c / m or *C-c * (`org-match-sparse-tree`)
: Create 作成します a sparse tree スパースツリーを with all headlines matching a tags search. With a *C-u* 前置引数といっしょだと ignore 無視します headlines that are not a TODO line.

M-x org-agenda m (`org-tags-view`)
: Create 作成します a global list of tag matches from all agenda files. 節 10.3.3 [Matching tags and properties], ページ 113, を見てください。。

M-x org-agenda M (`org-tags-view`)
: Create 作成します a global list of tag matches from all agenda files, but check only TODO items and force checking subitems（the オプションorg-tags-match-list-sublevels を見てください）。

These commands all prompt プロンプトを出します for a match string which allows 許可しています basic Boolean logic like '`+boss+urgent-project1`', to find entries with tags '`boss`' and '`urgent`', but not '`project1`', or '`Kathy|Sally`' to find entries which are tagged, like '`Kathy`' or '`Sally`'. The full syntax of the search string is rich and allows 可能にしています also matching against TODO keywords, entry levels and properties. For a complete description with many examples, 多くの例がついた see 節 10.3.3 [Matching tags and properties], ページ 113, を見てください。

7 Properties and Columns

A property is a key-value pair associated with an entry. Properties プロパティを can be set 設定することができます so they are associated with a single entry, with every entry in a tree, or with every entry in an Org file.

There are two main applications for properties in Org-mode. First, 最初に properties are like tags, but with a value. Imagine maintaining a file where you document bugs and plan releases for a piece of software. Instead かわりに of using 使う tags タグを like '`release_1`', '`release_2`', you can use 使うことができます a property, プロパティを say 例えば'`Release`' のような that in different subtrees has 持つ different values, 異なる値を such as '`1.0`' や'`2.0`' といった Second, 2 番目に you can use 使うことができます properties プロパティを to implement 実装するために(very basic) database capabilities（とても基本的な）データベース機能を in an Org buffer. Imagine keeping track of your music CDs, where properties could be things such as the album, artist, date of release, number of tracks, and so on.

Properties can be conveniently edited and viewed in column view（節 7.5 [Column View], ページ 70, を見てください）.

7.1 Property Syntax

Properties are key--value pairs. When they それが are associated 関連づけられているときには with a single entry 1 つのエントリーか or with a tree 1 つのツリーに they need to be inserted 挿入さる必用がありま す into a special drawer（節 2.8 [Drawers], ページ 16, を見てください）with the name '`PROPERTIES`', which これは has to be located ある必用がります right below a headline, and its planning line（節 8.3 [Deadlines and Scheduling], ページ 78, を見てください）when applicable. Each property is specified on a single line, with the key---surrounded by colons---first, and the value after it. Keys are case-insensitive. これは例です:

```
* CD collection
** Classic
*** Goldberg Variations
    :PROPERTIES:
    :Title:     Goldberg Variations
    :Composer:  J.S. Bach
    :Artist:    Glenn Gould
    :Publisher: Deutsche Grammophon
    :NDisks:    1
    :END:
```

Depending on 依存して、the value 値に of `org-use-property-inheritance` の a property プロパティは set this way is associated either with a single entry, or with the sub-tree defined by the entry, see 節 7.4 [Property Inheritance], ページ 69.

You may define 定義することができあますう the allowed values 許可されている値を for a particular property '`Xyz`' by setting 設定することによって a property '`Xyz_ALL`'. This special property is *inherited*, so if you set it in a level 1 entry, it applies to the entire tree. When allowed values 許可されている値が are defined, 定義されているときには、setting 設定は the corresponding property 対応するプロパティの becomes easier より簡単になり

and is less prone to typing errors. For the example 例については with the CD collection, we can pre-define publishers and the number of disks in a box like this: このようにして

```
* CD collection
  :PROPERTIES:
  :NDisks_ALL:  1 2 3 4
  :Publisher_ALL: "Deutsche Grammophon" Philips EMI
  :END:
```

If you want to set properties that can be inherited by any entry in a file, use a line like:

```
#+PROPERTY: NDisks_ALL 1 2 3 4
```

If you want to add to the value of an existing property, append a '`+`' to the property name. The following results in the property '`var`' having the value '`foo=1 bar=2`'.

```
#+PROPERTY: var  foo=1
#+PROPERTY: var+ bar=2
```

It is also possible 可能です to add to the values of inherited properties. The following results in the '`Genres`' property having the value '`Classic Baroque`' under the '`Goldberg Variations`' subtree.

```
* CD collection
** Classic
    :PROPERTIES:
    :Genres: Classic
    :END:
*** Goldberg Variations
    :PROPERTIES:
    :Title:     Goldberg Variations
    :Composer:  J.S. Bach
    :Artist:    Glenn Gould
    :Publisher: Deutsche Grammophon
    :NDisks:    1
    :Genres+:   Baroque
    :END:
```

Note 注意してください that a property can only have one entry per drawer.

Property values set with the global variable グローバル変数`org-global-properties` を使って設定されるプロパティは、can be inherited by all entries in all Org files.

The following commands 次のコマンドが help to work with properties: プロパティで作業することを助けてくれます:

M-TAB (`pcomplete`)
: After an initial colon in a line, complete property keys. All keys used in the current file are offered as possible completions. 可能な補完として

C-c C-x p (`org-set-property`)
: Set a property. プロパティを設定します。This prompts プロンプトを出します for a property name and a value. If necessary, 必用であれば the property drawer プロパティドロワーが is created 作成されます as well.

C-u M-x org-insert-drawer
: Insert 挿入します a property drawer into the current entry. The drawer is inserted early in the entry, but after the lines with planning information like deadlines.

C-c C-c (`org-property-action`)
: With point ポイントを in a property drawer, this executes property commands.

C-c C-c s (`org-set-property`)
: Set a property in the current entry. Both the property and the value can be inserted 挿入することができまう using completion. 補完を使って

S-RIGHT (`org-property-next-allowed-values`)
S-LEFT (`org-property-previous-allowed-value`)
: Switch 切り替えます property at point to the next/previous allowed value. 次/前の許可されている値に

C-c C-c d (`org-delete-property`)
: Remove a property from the current entry.

C-c C-c D (`org-delete-property-globally`)
: Globally remove a property, from all entries in the current file.

C-c C-c c (`org-compute-property-at-point`)
: Compute the property at point, using the operator and scope from the nearest column format definition.

7.2 Special Properties

Special properties provide an alternative access method to Org-mode features, like the TODO state or the priority of an entry, discussed in the previous chapters. This interface exists so that you can include 含めることができるように these states in a column view （節 7.5 [Column View], ページ 70, を見てください）, or to use them in queries. The following property names are special and should not be used 使われるべきではありません as keys キーとして in the properties drawer: プロパティドロワーの中の

'ALLTAGS'	All tags, including inherited ones.
'BLOCKED'	t if task is currently blocked by children or siblings.
'CATEGORY'	The category of an entry.
'CLOCKSUM'	The sum of CLOCK intervals in the subtree. `org-clock-sum` must be run first to compute the values in the current buffer.
'CLOCKSUM_T'	The sum of CLOCK intervals in the subtree for today. `org-clock-sum-today` must be run first to compute the values in the current buffer.
'CLOSED'	W いるこのエントリーがクローズにされたか？
'DEADLINE'	The deadline time string, without the angular brackets.
'FILE'	The filename the entry is located in.
'ITEM'	The headline of the entry.
'PRIORITY'	The priority of the entry, a string with a single letter.
'SCHEDULED'	The scheduling timestamp, without the angular brackets.
'TAGS'	The tags defined directly in the headline.

'TIMESTAMP' The first keyword-less timestamp in the entry.
'TIMESTAMP_IA' The first inactive timestamp in the entry.
'TODO' The TODO keyword of the entry.

7.3 Property Searches

To create 作成するには sparse trees スパースツリーを and special lists 特別なリストを with selection 選択がついた based on 基いた properties, プロパティに the same commands 同じコマンドが are used 使われます as for tag searches タグ検索と（節 6.4 [Tag Searches], ページ 64, を見てください）.

C-c / m or *C-c * (`org-match-sparse-tree`)
Create 作成します a sparse tree スパースツリーを with all matching entries. With a *C-u* 前置引数といっしょだと ignore 無視します headlines that are not a TODO line.

M-x org-agenda m, `org-tags-view`
Create 作成します a global list of tag/property matches from all agenda files.

M-x org-agenda M (`org-tags-view`)
Create 作成します a global list グローバルなリストを of tag matches from all agenda files, but check only TODO items and force checking of subitems（オプション`org-tags-match-list-sublevels` を見てください）。

The syntax for the search string is described in 節 10.3.3 [Matching tags and properties], ページ 113.

There is also a special command for creating sparse trees based on a single property:

C-c / p Create 作成します a sparse tree スパースツリーを based on the value of a property. This これは最初に first prompts プロンプトを出します for the name 名前を求めて of a property, and then 次に for a value. 値を求めて A sparse tree スパースツリーが is created 作成されます with all entries that define this property with the given value. If you enclose the value in curly braces, it is interpreted as a regular expression and matched against the property values.

7.4 Property Inheritance

The outline structure of Org documents lends itself to an inheritance model of properties: if the parent n a tree has a certain property, the children can inherit this property. Org-mode does not turn this on by default, デフォルトで because it can slow down property searches significantly かなり and is often not needed. However, しかし if you find inheritance useful, 継承が役に立つことがわかったら you can turn it on by setting 設定することによって the 変数`org-use-property-inheritance` を It may be set to `~t ~` to make all properties inherited from the parent, to a list of properties that should be inherited, 継承されるべき or to a regular expression that matches inherited properties. If a property has the value `nil`, this is interpreted as an explicit un-define of the property, so that inheritance search stops at this value and returns `nil`.

Org-mode has a few properties for which inheritance is hard-coded, at least for the special applications for which they are used:

`COLUMNS` The '`COLUMNS`' プロパティは defines 定義します the format of column view（節 7.5 [Column View], ページ 70, を見てください）。It is inherited in the sense that the level where a '`COLUMNS`' property is defined is used as the starting point for a column view table, independently of the location in the subtree from where columns view is turned on.

`CATEGORY` For agenda view, a category set through a '`CATEGORY`' property applies to the entire subtree.

`ARCHIVE` For archiving, the '`ARCHIVE`' property may define the archive location for the entire subtree（節 9.6.1 [Moving subtrees], ページ 105, を見てください）。

`LOGGING` The '`LOGGING`' property may define logging settings for an entry or a subtree（節 5.3.2 [Tracking TODO state changes], ページ 56, を見てください）。

7.5 Column View

A great way to view and edit properties in an outline tree is *column view*. In column view, each outline node is turned into a table row. Columns in this table provide access to properties of the entries. Org-mode implements columns by overlaying a tabular structure over the headline of each item. While the headlines have been turned into a table row, you can still change the visibility of the outline tree. For example, 例えば you get a compact table by switching to "contents" view—*S-TAB S-TAB*, or simply 単純に*c* while column view is active---but you can still open, read, and edit the entry below each headline. Or, または you can switch to column view after executing a sparse tree command and in this way get a table only for the selected items. Column view also works in agenda buffers（章 10 [Agenda Views], ページ 108, を見てください）。where queries have collected selected items, possibly from a number of files.

7.5.1 Defining columns

Setting up a column view first requires defining the columns. This is これは done 行なれます by defining 定義することによって a column format line. 列フォーマット行を

7.5.1.1 Scope of column definitions

To define 定義するにっは a column format 列フォーマットを for an entire file, use 使ってください a line like: このような行を

```
#+COLUMNS: %25ITEM %TAGS %PRIORITY %TODO
```

To specify 指定するには a format フォーマットを that only applies 適用される to a specific tree, 特定のツリーに対してだけ add 追加してください a '`COLUMNS`' プロパティを to the top node トップノードに対して of that tree, そのツリーの for example: 例えばこのようにして

```
** Top node for columns view
   :PROPERTIES:
   :COLUMNS: %25ITEM %TAGS %PRIORITY %TODO
   :END:
```

If a '`COLUMNS`' property is present in an entry, it defines 定義します columns for the entry itself, and for the entire subtree below it. Since the column definition is part of the hierarchical structure of the document, you can define columns on level 1 that are general

enough for all sublevels, and more specific columns further down, when you edit a deeper part of the tree.

7.5.1.2 Column attributes

A column definition sets 設定します the attributes 属性を of a column. The general definition looks like this: このように見えます:

```
%[WIDTH]PROPERTY[(TITLE)][{SUMMARY-TYPE}]
```

Except for the percent sign and the property name, all items are optional. オプションです The individual parts have the following meaning:

WIDTH An integer specifying the width of the column in characters. If omitted, 省略されたら the width この幅は is determined automatically. 自動的に決定されます

PROPERTY
The property プロパティです that should be edited 編集されるべき in this column. この列の中で Special properties representing meta data are allowed 許可されています here ここで as well（節 7.2 [Special Properties], ページ 68, を見てください）。

TITLE The header text for the column. If omitted, the property name is used.

SUMMARY-TYPE
The summary type. If specified, the column values for parent nodes are computed from the children[1].

Supported summary types are:

'`+`'	Sum numbers in this column.
'`+;%.1f`'	'`+`' に似ていますが but format result with '`%.1f`'.
'`$`'	Currency, short for '`+;%.2f`'.
'`min`'	Smallest number in column.
'`max`'	Largest number.
'`mean`'	Arithmetic mean of numbers.
'`X`'	Checkbox status, '`[X]`' if all children are '`[X]`'.
'`X/`'	Checkbox status, '`[n/m]`'.
'`X%`'	Checkbox status, '`[n%]`'.
'`:`'	Sum times, HH:MM, plain numbers are minutes.
'`:min`'	Smallest time value in column.
'`:max`'	Largest time value.
'`:mean`'	Arithmetic mean of time values.
'`@min`'	Minimum age[2] (in days/hours/mins/seconds).
'`@max`'	Maximum age (in days/hours/mins/seconds).
'`@mean`'	Arithmetic mean of ages (in days/hours/mins/seconds).
'`est+`'	Add low-high estimates.

You can also define custom summary types by setting 設定することによって`org-columns-summary-types` を

[1] If more than one summary type applies to the same property, the parent values are computed according 従って to the first of them.

[2] An age can be defined as a duration, using units defined in `org-duration-units`, e.g., 例えば'`3d 1h`'. If any value in the column is as such, the summary is also expressed as a duration.

The ‘`est+`’ summary type requires further explanation. It is used for combining estimates, expressed as low-high ranges. For example, 例えば instead かわりに of estimating 見つもる a particular task will take 5 days, you might estimate it as 5--6 days if you're fairly confident you know how much work is required, or 1--10 days if you do not really know what needs to be done. 行なわれる必用があること Both ranges average at 5.5 days, but the first represents a more predictable delivery.

When combining 組み合わせるときには a set of such estimates, simply 単純に adding 合計すると the lows and highs produces an unrealistically wide result. Instead, かわりに‘`est+`’ は adds the statistical mean and variance of the subtasks, generating a final estimate from the sum. For example, 例えば suppose しましょう you had ten tasks, each of which was estimated at 0.5 to 2 days of work. Straight addition produces an estimate of 5 to 20 days, representing what to expect if everything goes either extremely well or extremely poorly. In contrast, ‘`est+`’ estimates the full job more realistically, at 10--15 days.

Here これは is an example 例ですう for a complete columns definition, along with ともに allowed values 許可されている値と

7.5.2 Using column view

Turning column view on or off

C-c C-x C-c (`org-columns`)
: Turn on column view. If point ポイントが is before the first headline in the file, column view is turned on for the entire file, using the ‘`#+COLUMNS`’ definition. If point ポイントが is somewhere どこかにあれば inside the outline, this command searches the hierarchy, up from point, for a ‘`COLUMNS`’ property that defines 定義します a format. When one is found, 1 つ見つかったら、the column view table is established for the tree starting at the entry that contains 含んでいます the ‘`COLUMNS`’ property. If no such property is found, the format is taken from the ‘`#+COLUMNS`’ line or from the variable 変数`org-columns-default-format`, and column view is established for the current entry and its subtree.

r or *g* (`org-columns-redo`)
: Recreate the column view, to include recent changes made in the buffer.

q (`org-columns-quit`)
: Exit column view.

Editing values

`LEFT`, `RIGHT`, `UP`, `DOWN`
: Move 移動します through the column view from field to field.

1..9,0
: Directly select the Nth allowed value, N 番目の許可されている値を*0* selects the 10th value. 10 番目の値を

n or *S-RIGHT* (`org-columns-next-allowed-value`)
p or *S-LEFT* (`org-columns-previous-allowed-value`)
: Switch to the next/previous allowed value of the field. For this, you have to have specified 指定しておく必用があります allowed values 許可されている値を for a property.

`e` (`org-columns-edit-value`)
: Edit the property at point. ポイントにあるプロパティを編集します。For the special properties, this invokes the same interface that you normally 通常は use 使う to change 変更するために that property. そのプロパティを For example, 例えば the tag completion or fast selection interface pops up when editing a 'TAGS' property.

C-c C-c (`org-columns-set-tags-or-toggle`)
: ポイントにチェックボックスがあれば、それをトグルします。

`v` (`org-columns-show-value`)
: View the full value of this property. This is useful これが役に立ちます if the width of the column is smaller than that of the value.

`a` (`org-columns-edit-allowed`)
: Edit the list of allowed values for this property. If the list is found 見つかったら in the hierarchy, the modified values is stored there. If no list is found, the new value is stored in the first entry that is part of the current column view.

Modifying column view on-the-fly

`<` (`org-columns-narrow`)
`>` (`org-columns-widen`)
: Make the column narrower/wider by one character.

S-M-RIGHT (`org-columns-new`)
: Insert 挿入します a new column, to the left of the current column.

S-M-LEFT (`org-columns-delete`)
: Delete the current column.

7.5.3 Capturing column view

Since column view is just an overlay over a buffer, it cannot be exported or printed directly. If you want to capture a column view, use a '`columnview`' dynamic block（節 A.6 [Dynamic Blocks], ページ 263, を見てください）。The frame of this block このブロックのフレームは looks like this: このにょうに見めます

```
* The column view
#+BEGIN: columnview :hlines 1 :id "label"

#+END:
```

This dynamic block has the following parameters:

'`:id`'
: This これは is the most important parameter. 最も重要なパラメータです Column view is a feature 機能です that is often localized to a certain (sub)tree, and the capture block might be at a different location in the file. To identify the tree ツリーを特定するには、whose view to capture, ビューをキャプチャすべき you can use four values: 4 つの値を使うことができます:

  '`local`'
  : Use the tree in which the capture block is located.

  '`global`'
  : Make a global view, including 含む all headings in the file.

‘`file:FILENAME`’
: Run column view at the top of the *FILENAME* file.

‘`LABEL`’
: Call column view in the tree that has an ‘`ID`’ property with the value *LABEL*. You can use 使うことができます*M-x org-id-copy* を to create 作成するために a globally unique グローバルにユニークな ID を for the current entry and copy it to the kill-ring.

‘`:hlines`’
: When `t` のときには、insert 挿入します an hline を after every line. 全ての行の後に When a number 数値の N のときには insert 挿入します an hline before each headline with level `<= N`.

‘`:vlines`’
: When non-`nil`, `nil` でない値のときには、force 強制します column groups to get vertical lines.

‘`:maxlevel`’
: When set to a number, 数値に設定されているときには、do not capture キャプチャしません entries below this level.

‘`:skip-empty-rows`’
: When non-`nil`, `nil` でない値のときには skip スキップします rows 行を where the only non-empty specifier of the column view is ‘`ITEM`’.

‘`:exclude-tags`’
: List of tags to exclude from column view table: entries with these tags will be excluded from the column view.

‘`:indent`’
: When non-`nil`, `nil` でない値のときには indent インデントします each ‘`ITEM`’ field according 従って to its level. そのレベルに従って

‘`:format`’
: Specify a column attribute（節 7.5.1.2 [Column attributes], ページ 70, を見てください）for the dynamic block.

The following commands insert or update the dynamic block:

C-c C-x i (`org-insert-columns-dblock`)
: Insert 挿入します a dynamic block capturing キャプチャしている a column view. Prompt for the scope or ID of the view.

C-c C-c C-c C-x C-u (`org-dblock-update`)
: Update dynamic block at point. point ポイントが needs to be in the ‘`#+BEGIN`’ line of the dynamic block. 動的ブロックの

C-u C-c C-x C-u (`org-update-all-dblocks`)
: Update all dynamic blocks 全ての動的ブロックを更新します（節 A.6 [Dynamic Blocks], ページ 263, を見てください）。This is useful これが役に立ちます if you have several clock table blocks, column-capturing blocks or other dynamic blocks in a buffer.

You can add 追加することができます formulas 数式を to the column view table and you may add plotting instructions in front of the table---these survive an update of the block. If there is a ‘`TBLFM`’ keyword after the table, the table is recalculated automatically 自動的に再計算されます after an update. 更新の後で

An alternative way to capture and process property values into a table is provided by Eric Schulte's '`org-collector.el`', which これは is a contributed package 寄贈パッケージです

8 Dates and Times

To assist 助けるために project planning, プロジェクト計画を TODO items TODO アイテムに can be labeled ラベルをつけることができます with a date and/or a time. The specially formatted string carrying the date and time information is called a *timestamp* in Org-mode. This may be a little confusing because timestamp is often used as indicating when いつ something was created or last changed. However, しかし in Org-mode では this term この用語は is used in a much wider sense. ずっと広い意味で使われています

8.1 Timestamps, Deadlines and Scheduling

A timestamp is a specification of a date (possibly with a time or a range of times) in a special format, either '`<2003-09-16 Tue>`' or '`<2003-09-16 Tue 09:39>`' or '`<2003-09-16 Tue 12:00-12:30>`'[1]. A timestamp can appear anywhere どこにでも in the headline or body of an Org tree entry. Its presence causes entries to be shown 表示させます on specific dates in the agenda（節 10.3.1 [Weekly/daily agenda], ページ 110, を見てください）。We distinguish:

Plain timestamp; Event; Appointment
: A simple timestamp just assigns a date/time to an item. This is just like writing down an appointment or event in a paper agenda. In the agenda display, the headline of an entry associated with a plain timestamp is shown exactly on that date.

```
* Meet Peter at the movies
  <2006-11-01 Wed 19:15>
* Discussion on climate change
  <2006-11-02 Thu 20:00-22:00>
```

Timestamp with repeater interval
: A timestamp may contain a *repeater interval*, indicating that it applies not only on the given date, but again and again after a certain interval of N days (d), weeks (w), months (m), or years (y). The following shows up 出現します in the agenda every Wednesday:

```
* Pick up Sam at school
  <2007-05-16 Wed 12:30 +1w>
```

Diary-style sexp entries
: For more complex date specifications, Org-mode supports サポートしています using the special sexp diary entries implemented in the Emacs calendar/diary package[2]. For example, 例えば with optional time:

```
* 22:00-23:00 The nerd meeting on every 2nd Thursday of the month
```

[1] The Org date format is inspired by the standard ISO 8601 date/time format. To use 使うには an alternative format, see 節 8.2.2 [Custom time format], ページ 77. The day name is optional when you type the date yourself. However, any date inserted or modified by Org adds that day name, for reading convenience.

[2] When working with the standard diary sexp functions, you need to be very careful with the order of the arguments. That order depends evilly on the variable 変数`calendar-date-style`. For example, 例えば to specify a date December 12, 2005, the call might look like '`(diary-date 12 1 2005)`' or '`(diary-date 1 12 2005)`' or '`(diary-date 2005 12 1)`', depending on 依存して the settings. 設定に This has been the source of much confusion. Org-mode users can resort to special versions of these functions like `org-date`

```
<%%(diary-float t 4 2)>
```

Time/Date range
: Two timestamps connected by '`--`' denote a range. The headline is shown on the first and last day of the range, and on any dates that are displayed and fall in the range. これは例です:

```
** Meeting in Amsterdam
   <2004-08-23 Mon>--<2004-08-26 Thu>
```

Inactive timestamp
: Just like a plain timestamp, but with square brackets instead かわりに of angular ones. 角括弧の These timestamps are inactive in the sense that they do *not* trigger an entry to show up in the agenda.

```
* Gillian comes late for the fifth time
  [2006-11-01 Wed]
```

8.2 Creating Timestamps

For Org-mode to recognize timestamps, they need to be in the specific format. All commands listed below produce timestamps in the correct format.

C-c . (`org-time-stamp`)
: Prompt プロンプトを出します for a date and insert a corresponding timestamp. When point ポイントが is at an existing timestamp in the buffer, the command このコマンドは is used 使われます to modify this timestamp このタイムスタンプを instead かわりに of inserting 挿入する a new one. 新しいタイムスタンプを When this command このコマンドが is used twice 2 回 in succession, 続けて使われるときにっは、a time range is inserted. 時間の範囲が挿入されます。

  When called with a prefix argument, 1 つの前置引数といっしょに呼ばれるときには、use 使います the alternative format 代替のフォーマットを which contains 含んでいる date and time. 日付と時間を The default time デフォルトの時間を can be rounded 丸まることができます to multiples of 5 minutes. 5 分の倍数にオプション`org-time-stamp-rounding-minutes` を見てください。

  With two prefix arguments, 2 つの前置引数といっしょだと、insert 挿入しますう an active timestamp アクティブなタイムスタンプを with the current time without prompting. プロンプトを出すことなく

C-c ! (`org-time-stamp-inactive`)
: Like *C-c .* に似ていますが、but insert 挿入します an inactive timestamp 非アクティブなタイムスタンプを that does not cause an agenda entry.

C-c C-c Normalize timestamp, insert or fix day name if missing or wrong.

C-c < (`org-date-from-calendar`)
: Insert 挿入します a timestamp corresponding to point date in the calendar.

C-c > (`org-goto-calendar`)
: Access the Emacs calendar for the current date. If there is a timestamp in the current line, go to the corresponding date instead. かわりに

---

or `org-anniversary`. These work just like the corresponding `diary-` functions, but with stable ISO order of arguments (year, month, day) wherever applicable, independent of the value of `calendar-date-style`.

C-c C-o (`org-open-at-point`)
: Access the agenda for the date given by the timestamp or -range at point（節 10.3.1 [Weekly/daily agenda], ページ 110, を見てください）。

S-LEFT (`org-timestamp-down-day`)
S-RIGHT (`org-timestamp-up-day`)
: Change date at point by one day. These key bindings conflict 衝突します with shift-selection and related modes（節 15.13.2 [Conflicts], ページ 252, を見てください）。

S-UP (`org-timestamp-up`)
S-DOWN (`org-timestamp-down`)
: On the beginning or enclosing bracket of a timestamp, change its type. その種類を変更します。Within a timestamp, change the item under point. Point can be on a year, month, day, hour or minute. When the timestamp タイムスタンプが contains 含んでいるときには a time range 時間の範囲を like '`15:30-16:30`' のような modifying 変更すると the first time 最初の時間を also shifts the second, shifting シフトします the time block この時間のブロックを with constant length. 一定の長さ To change the length, 長さを変更するには、modify the second time. 2 番目の時間を変更してください。Note 注意してください that if point is in a headline and not at a timestamp, these same keys modify the priority 優先順位を of an item（節 5.4 [Priorities], ページ 58, を見てください）。The key bindings これらのキーバインディングは also conflict 衝突します with shift-selection シフト選択と and related modes 関係するモードと（節 15.13.2 [Conflicts], ページ 252, を見てください）。

C-c C-y (`org-evaluate-time-range`)
: Evaluate a time range by computing the difference between start and end. With a prefix argument, 1 つの前置引数といっしょだと、insert 挿入します result 結果を after the time range 時間の範囲の後に(in a table: into the following column).

8.2.1 The date/time prompt

When Org-mode が prompts for a date/time, 日付/時間を求めてプロンプトを出すときには、the default is shown in default date/time format, and the prompt therefore よって seems to ask for a specific format. But しかし it in fact accepts date/time information in a variety of formats. Generally, the information should start はじまるべき at the beginning はじめで of the string. Org-mode finds whatever information is in there そこにある and derives anything you have not specified from the *default date and time*. The default デフォルトは is usually 通常は the current date and time, 現在の日付と時間ですが、but when modifying 変更するときには an existing timestamp, 既存のタイムスタンプを or when entering the second stamp of a range, it is taken from the stamp in the buffer. When filling in information, 情報を埋めるときには、Org-mode は assumes 仮定します that most of the time you want to enter a date in the future: if you omit the month/year and the given day/month is *before* today, it assumes that you mean a future date

8.2.2 Custom time format

Org-mode は uses 使います the standard ISO notation for dates and times as it is defined in ISO 8601. If you cannot get used to this and require another representation of date and

time to keep you happy, you can get 得ることができます it それを by customizing カスタマイズすることによって the variables 変数`org-display-custom-times` と`org-time-stamp-custom-formats` を

C-c C-x C-t (`org-toggle-time-stamp-overlays`)
: Toggle トグルします the display of custom formats for dates and times.

Org-mode needs 必用ですう the default format for scanning, so the custom date/time format does not *replace* the default format. Instead, かわりに it それは is put *over* the default format using text properties. This has the following consequences:

- You cannot place point onto a timestamp anymore, only before or after.
- The *S-UP* and *S-DOWN* keys can no longer be used to adjust each component of a timestamp. If point ポイントが is at the beginning of the stamp, *S-UP* and *S-DOWN* change the stamp by one day, just like *S-LEFT S-RIGHT*. At the end of the stamp, change the time by one minute.
- If the timestamp contains 含んでいます a range of clock times or a repeater, these are not overlaid, but remain in the buffer as they were.
- When you delete 削除するときには a timestamp character-by-character, it only disappears from the buffer after *all* (invisible) characters belonging to the ISO timestamp have been removed.
- If the custom timestamp format is longer than the default and you are using dates in tables, table alignment will be messed up. If the custom format is shorter, things do work as expected.

8.3 Deadlines and Scheduling

A timestamp may be preceded by special keywords to facilitate planning. Both the timestamp and the keyword have to be positioned immediately after the task they refer to.

'`DEADLINE`'
: Meaning: the task (most likely a TODO item, though not necessarily) is supposed to be finished on that date.

  On the deadline date, the task is listed in the agenda. In addition, 加えて the agenda for *today* carries a warning about the approaching or missed deadline, starting `org-deadline-warning-days` before the due date, and continuing until the entry is marked DONE. An example:

```
*** TODO write article about the Earth for the Guide
    DEADLINE: <2004-02-29 Sun>
    The editor in charge is [[bbdb:Ford Prefect]]
```

  You can specify a different lead time for warnings for a specific deadlines using the following syntax. Here これは is an example 例です with a warning period of 5 days '`DEADLINE: <2004-02-29 Sun -5d>`'. This warning is deactivated if the task gets scheduled and you set `org-agenda-skip-deadline-prewarning-if-scheduled` to `t`.

'`SCHEDULED`'
: Meaning: you are planning to start working on that task on the given date.

The headline is listed under the given date[3]. In addition, 加えて a reminder that the scheduled date has passed is present in the compilation for *today*, until the entry is marked DONE とマークされるまで i.e., すなわち the task このタスクは is automatically 自動的にフォワードされます forwarded until completed. 完了されるまで

```
*** TODO Call Trillian for a date on New Years Eve.
    SCHEDULED: <2004-12-25 Sat>
```

If you want to *delay* the display of this task in the agenda, use '`SCHEDULED: <2004-12-25 Sat -2d>`': the task is still scheduled on the 25th but will appear two days later. In case the task contains 含んでいます a repeater, the delay is considered to affect all occurrences; if you want the delay to only affect the first scheduled occurrence of the task, use '`--2d`' instead. かわりに See 見てくださいorg-scheduled-delay-days とorg-agenda-skip-scheduled-delay-if-deadline を for details 詳細については on how 方法についての to control コントロールする this globally or per agenda.

> **Important:** Scheduling an item in Org-mode should *not* be understood 理解されるべきでは/ありません in the same way that we understand *scheduling a meeting*. Setting a date for a meeting is just a simple appointment, you should mark マークするべきです this entry with a simple plain timestamp, to get this item shown on the date where it applies. This is a frequent misunderstanding by Org users. In Org-mode, *scheduling* means setting a date when you want to start working on an action item.

You may use timestamps with repeaters in scheduling and deadline entries. Org-mode issues early and late warnings based on the assumption that the timestamp represents the *nearest instance* of the repeater. However, しかし the use 使用は of diary S-exp entries like

```
<%%(diary-float t 42)>
```

in scheduling and deadline timestamps is limited. Org-mode does not know enough about the internals of each S-exp function to issue early and late warnings. However, しかし it それっは shows 表示します the item on each day where the S-exp entry matches. マッチする

8.3.1 Inserting deadlines or schedules

The following commands 続くコマンドが allow 可能にします you to quickly 素早く insert 挿入することを a deadline 締め切りを or to schedule スケジュールすることを an item: アイテムを

8.3.2 Repeated tasks

Some tasks need to be repeated again and again. Org-mode helps to organize such tasks using a so-called repeater in a '`DEADLINE`', '`SCHEDULED`', or plain timestamps[4]. In the following example:

```
** TODO Pay the rent
```

[3] It will still be listed on that date after it has been marked DONE. If you do not like this, これが好きでなかったら」set 設定してください。the variable 変数org-agenda-skip-scheduled-if-done を

[4] Org does not repeat inactive timestamps, however. See 節 8.1 [Timestamps], ページ 75.

```
DEADLINE: <2005-10-01 Sat +1m>
```

the '+1m' は is a repeater; リピーターです the intended interpretation 意図されている解釈方法は is that the task has a deadline 締め切りが on '<2005-10-01>' に and repeats すあ itself every (one) month starting from that time. その時間から You can use 使うことができます yearly, 年ごと、月ごと、monthly, weekly, daily 日ごと、and hourly repeat cookies 時間ごとの繰り返しのクッキーを by using the 'y' と'w' と'm' と'd' と'h' という文字を使うことによって、If you need both a repeater and a special warning period in a deadline entry, the repeater should come first 最初にきて and the warning period last 最後にくるべきです

```
DEADLINE: <2005-10-01 Sat +1m -3d>
```

Deadlines and scheduled items produce entries in the agenda when they are over-due, so it is important to be able to mark such an entry as DONE once you have done so. When you mark マークしたら a 'DEADLINE' or a 'SCHEDULED' with the TODO keyword 'DONE', it no longer produces entries in the agenda. The problem with this is, however, しかし is that then also the *next* instance of the repeated entry will not be active. Org-mode deals with this in the following way: when you try to mark such an entry DONE, using 使って*C-c C-t* を it shifts the base date of the repeating timestamp by the repeater interval, and immediately すぐに sets 設定します the entry state back to TODO[5]. In the example above, setting the state to DONE would actually switch the date like this:

```
** TODO Pay the rent
   DEADLINE: <2005-11-01 Tue +1m>
```

To mark マークするには a task タスクを with a repeater リピーターを持つ as DONE として use *C-- 1 C-c C-t* を使ってください。i.e., すなわちorg-todo です with a numeric prefix argument of '-1' という前置引数といっしょの

A timestamp[6] is added under the deadline, to keep a record that you actually acted on the previous instance of this deadline.

As a consequence of shifting the base date, this entry is no longer もは visible in the agenda when checking past dates, but all future instances will be visible. 可視になります

With the '+1m' というクッキーを使うと the date shift is always 常に exactly one month. So if you have not paid the rent for three months, marking this entry DONE still keeps it as an overdue deadline. Depending on 依存して the task, タスクに this これは may not be the best way to handle it. For example, 例えば if you forgot to call your father for 3 weeks, it does not make sense to call him 3 times in a single day to make up for it. それを埋め合わせるために Finally 最後に there are tasks タスクがあります like changing 変更することのような batteries バッテリを which should always 常に repeat 繰り返すべき a certain time *after* the last time you did it. For these tasks, Org-mode has special repeaters '++' and '.+'. 例です:

```
** TODO Call Father
   DEADLINE: <2008-02-10 Sun ++1w>
   Marking this DONE shifts the date by at least one week, but also
```

[5] In fact, the target state is taken from, in this sequence, the 'REPEAT_TO_STATE' property, the variable 変数org-todo-repeat-to-state if it is a string, the previous TODO state if org-todo-repeat-to-state is t, or the first state of the TODO state sequence.

[6] You can change 変更することができます this これを using the option org-log-repeat, or the 'STARTUP' options 'logrepeat', 'lognoterepeat', and 'nologrepeat'. With 'lognoterepeat', you will also be prompted for a note.

```
   by as many weeks as it takes to get this date into the future.
   However, しかし
   it それは
   stays on a Sunday,
   even if you called and marked it
   done on Saturday.

** TODO Empty kitchen trash
   DEADLINE: <2008-02-08 Fri 20:00 ++1d>
   Marking this DONE shifts the date by at least one day, and also
   by as many days as it takes to get the timestamp into the future.
   Since there is a time in the timestamp, the next deadline in the
   future will be on today's date if you complete the task before
   20:00.

** TODO Check the batteries in the smoke detectors
   DEADLINE: <2005-11-01 Tue .+1m>
   Marking this DONE will shift the date to one month after today.
```

You may have both scheduling and deadline information for a specific task. If the repeater is set for the scheduling information only, you probably want the repeater to be ignored after the deadline. If so, そうであれば、set 設定してください the variable 変数`org-agenda-skip-scheduled-if-deadline-is-shown` を`repeated-after-deadline` に However, しかし any scheduling information without a repeater リピーターのない is no longer relevant once the task is done, and thus, よって removed upon repeating the task. If you want both scheduling and deadline information to repeat after the same interval, set the same repeater for both timestamps.

An alternative to using a repeater is to create a number of copies of a task subtree, with dates shifted in each copy. The command *C-c C-x c* was created for this purpose; it is described in 節 2.5 [Structure Editing], ページ 9.

8.4 Clocking Work Time

Org-mode allows 可能にしえちます you to clock the time you spend on specific tasks in a project. When you start はじめるときには working on an item, you can start the clock. When you stop 止めるときか working on that task, or when you mark the task done, the clock is stopped and the corresponding time interval is recorded. 記録されます It also computes the total time spent on each subtree[7] of a project. And そしえ it remembers a history or tasks recently clocked, to that you can jump quickly 素早く between a number of tasks absorbing your time.

To save 保存するには the clock history 時間の計測の履歴を across Emacs sessions, use: 次を使ってください

```
(setq org-clock-persist 'history)
(org-clock-persistence-insinuate)
```

[7] Clocking only works if all headings are indented with less than 30 stars. This is a hard-coded limitation of `lmax` in `org-clock-sum`.

When you clock into クロックインするときには a new task 新しいタスクに after resuming Emacs, the incomplete clock[8] is retrieved（[Resolving idle time (1)], ページ 87, を見てください）and you are prompted プロンプトが出ます about what to do with it.

8.4.1 Clocking commands

C-c C-x C-i (`org-clock-in`)

Start the clock on the current item (clock-in). This inserts 挿入します the CLOCK keyword together with a timestamp. If this is not the first clocking of this item, the multiple CLOCK lines are wrapped into a '`LOGBOOK`' drawer（変数`org-clock-into-drawer` も見てください）。You can also overrule the setting of this variable この変数の for a subtree by setting 設定することによって a '`CLOCK_INTO_DRAWER`' プロパティか'`LOG_INTO_DRAWER`' プロパティを When called with a *C-u* 前置引数といっしょに呼ばれるときには、prefix argument, select the task from a list of recently clocked tasks. With two *C-u C-u* prefixes, clock into the task at point and mark it as the default task; the default task is always 常に be available with letter *d* when selecting a clocking task. With three *C-u C-u C-u* prefixes, force continuous clocking by starting the clock when the last clock stopped.

While the clock is running, Org Org-mode は shows 表示します the current clocking time in the mode line, along with ともに the title タイトルと of the task. そのタスクの The clock time shown is all time ever clocked for this task and its children. If the task has an effort estimate（節 8.5 [Effort Estimates], ページ 89, を見てください）。the mode line displays 表示します the current clocking time against it[9]. If the task is a repeating one（節 8.3.2 [Repeated tasks], ページ 80, を見てください）、show only the time since the last reset of the task[10]. You can exercise more control over show time with the '`CLOCK_MODELINE_TOTAL`' property. It may have the values '`current`' to show only the current clocking instance, '`today`' to show all time clocked on this tasks today ---see also the variable 変数`org-extend-today-until`, `all` to include all time, or `auto` which is the default[11]. Clicking with *mouse-1* onto the mode line entry pops up a menu with clocking options.

C-c C-x C-o (`org-clock-out`)

Stop the clock (clock-out). This inserts 挿入します another timestamp at the same location where the clock was last started. It also directly computes the resulting time in inserts 挿入します it それを after the time range as '`=>HH:MM`'. See 見てください。the 変数`org-log-note-clock-out` を for the possibility 可能性につて to record 記録する an additional note 追加のノートを together with ともに the clock-out timestamp[12].

---

[8] To resume 再開するには the clock under the assumption that you have worked on this task while outside Emacs, use '`(setq org-clock-persist t)`'.

[9] To add 追加するには an effort estimate "on the fly", hook a function doing this to `org-clock-in-prepare-hook`.

[10] The last reset of the task is recorded by the '`LAST_REPEAT`' プロパティによって記録されます。

[11] See also the variable 変数`org-clock-mode-line-total`.

[12] The corresponding in-buffer setting is: '`#+STARTUP: lognoteclock-out`'.

C-c C-x C-x (`org-clock-in-last`)
: Re-clock the last clocked task. With one *C-u* prefix argument, select the task from the clock history. With two *C-u* prefixes, force continuous clocking by starting the clock when the last clock stopped.

C-c C-x C-e (`org-clock-modify-effort-estimate`)
: Update the effort estimate for the current clock task.

C-c C-c or *C-c C-y* (`org-evaluate-time-range`)
: Recompute the time interval after changing one of the timestamps. This is only necessary if you edit the timestamps directly. If you change them with *S-<cursor>* keys, the update is automatic.

C-S-UP (`org-clock-timestamps-up`)
C-S-DOWN (`org-clock-timestamps-down`)
: On CLOCK log lines, increase/decrease both timestamps so that the clock duration keeps the same value.

S-M-UP (`org-timestamp-up`)
S-M-DOWN (`org-timestamp-down`)
: On 'CLOCK' log lines, increase/decrease the timestamp at point and the one of the previous, or the next, clock timestamp by the same duration. For example, 例えば if you hit *S-M-UP* to increase a clocked-out timestamp by five minutes, then the clocked-in timestamp of the next clock is increased by five minutes.

C-c C-t (`org-todo`)
: Changing the TODO state of an item to DONE automatically 自動的に stops the clock 時間の計測を if it is running in this same item.

C-c C-x C-q (`org-clock-cancel`)
: Cancel the current clock. This is useful これが役に立ちます if a clock was started by mistake, or if you ended up working on something else.

C-c C-x C-j (`org-clock-goto`)
: Jump ジャンプしまづ to the headline of the currently clocked in task. With a *C-u* 前置引数といっしょだしたお select 選択します the target task from a list of recently clocked tasks.

C-c C-x C-d (`org-clock-display`)
: Display 表示しまう time summaries for each subtree in the current buffer. This puts overlays at the end of each headline, showing the total time recorded under that heading, including 含む the time of any subheadings. You can use 使うことができます visibility cycling 可視性の切り替えを to study the tree, but the overlays オーバーレイは disappear 消えます when you change the buffer（変数`org-remove-highlights-with-change` を見てください）or press *C-c C-c* を押してください

The *l* key may be used 使うことができます in the agenda（節 10.3.1 [Weekly/daily agenda], ページ 110, を見てください）to show which tasks have been worked on or closed during a day.

Important: note that both `org-clock-out` and `org-clock-in-last` can have a global keybinding and do not modify the window disposition.

8.4.2 The clock table

Org-mode can produce quite complex reports based on the time clocking information. Such a report is called a *clock table*, because it is formatted as one or several Org tables.

You can insert, or update, a clock table through Org dynamic blocks insert command（節 A.6 [Dynamic Blocks], ページ 263, を見てください）、by pressing *C-c C-x x c l o c k t a b l e RET* を押すことににによって When called with a prefix argument, 1 つの前置引数といっしょに呼ばれるときにぬうあ jump ジャンプします to the first clock table in the current document and update it. The clock table includes archived trees.

C-c C-c or *C-c C-x C-u* (`org-dblock-update`)
: Update dynamic block at point. Point ポイントが needs to be in the '`BEGIN`' line of the dynamic block. 動的ブロックの

C-u C-c C-x C-u
: Update all dynamic blocks 全ての動的ブロックを更新します（節 A.6 [Dynamic Blocks], ページ 263, を見てください）。This is useful これが役に立ちます if you have あれば several clock table blocks in a buffer.

S-LEFT
S-RIGHT (`org-clocktable-try-shift`)
: Shift the current '`:block`' interval and update the table. Point ポイントが needs to be in the '`#+BEGIN: clocktable`' line for this command. このコマンドが動くためには If '`:block`' is '`today`', it is shifted to '`today-1`', etc.

Here これは is an example of the frame for a clock table as it is inserted into the buffer with the *C-c C-x C-r* command:

```
#+BEGIN: clocktable :maxlevel 2 :emphasize nil :scope file
#+END: clocktable
```

The '`#+BEGIN`' line and specify a number of options to define the scope, structure, and formatting of the report. Defaults デフォルトを for all these options can be configured 設定することができます in the variable 変数`org-clocktable-defaults` を

First there are options that determine which clock entries are to be selected:

'`:maxlevel`'
: Maximum level depth to which times are listed in the table. Clocks at deeper levels are summed into the upper level.

'`:scope`' The scope to consider. This can be any of the following:

'`nil`'	the current buffer or narrowed region
'`file`'	the full current buffer
'`subtree`'	the subtree where the clocktable is located
'`treeN`'	the surrounding level N tree, for example '`tree3`'
'`tree`'	the surrounding level 1 tree
'`agenda`'	all agenda files
'`("file" ...)`'	scan these files
'`FUNCTION`'	scan files returned by calling *FUNCTION* with no argument
'`file-with-archives`'	current file and its archives
'`agenda-with-archives`'	all agenda files, including archives

‘`:block`’ The time block to consider. This block is specified either absolutely, or relative 相対的に to the current time and may be any of these formats:

‘`2007-12-31`’	New year eve 2007
‘`2007-12`’	December 2007
‘`2007-W50`’	ISO-week 50 in 2007
‘`2007-Q2`’	2nd quarter in 2007
‘`2007`’	the year 2007
‘`today`’, ‘`yesterday`’, ‘`today-N`’	a relative day
‘`thisweek`’, ‘`lastweek`’, ‘`thisweek-N`’	a relative week
‘`thismonth`’, ‘`lastmonth`’, ‘`thismonth-N`’	a relative month
‘`thisyear`’, ‘`lastyear`’, ‘`thisyear-N`’	a relative year
‘`untilnow`’[13]	all clocked time ever

When this option このオプションが is not set, Org falls back 戻りまう to the value 値に in `org-clock-display-default-range` の中にある which これはデフォルトで defaults to the current year. 現在の年です

Use 使ってくださいS-LEFT かS-RIGHT を to shift シフトするには the time interval.

‘`:tstart`’ A time string specifying when to start considering times. Relative times 相対的な時間を like ‘`"<-2w>"`’ のような can also be used. 使うこともできます See 節 10.3.3 [Matching tags and properties], ページ 113, を見てください。for relative time syntax. 相対的な時間の文法については、

‘`:tend`’ A time string specifying 指定している when to stop considering times. Relative times 相対的な時間を like ‘`"<now>"`’ のような can also be used. 使うこともできます See 節 10.3.3 [Matching tags and properties], ページ 113, を見てください。for relative time syntax. 相対的な時間の文法については、

‘`:wstart`’ The starting day of the week. デフォルトは月曜日を意味する 1 です。

‘`:mstart`’ The starting day of the month. デフォルトは 1 日を意味する 1 です。

‘`:step`’ Set to ‘`day`’, ‘`week`’, ‘`month`’ or ‘`year`’ to split the table into chunks. To use this, これを使うには、either ‘`:block`’ か‘`:tstart`’ か‘`:tend`’ が are required. 必用でせいう

‘`:stepskip0`’
Do not show 表示しません steps that have zero time.

‘`:fileskip0`’
Do not show table sections from files which did not contribute.

‘`:match`’ A tags match to select entries that should contribute. 寄与すべき See 節 10.3.3 [Matching tags and properties], ページ 113, を見てください。for the match syntax. マッチの文法については、

Then there are options that determine 決定する the formatting of the table. There options are interpreted by the function `org-clocktable-write-default`, but you can specify 指定することができます your own function using the ‘`:formatter`’ parameter.

[13] When using 使うときには`:step`, `untilnow` starts from the beginning of 2003, not the beginning of time.

‘:emphasize’
When t のときには、emphasize 強調します level one and level two items.

‘:lang’ Language[14] to use for descriptive cells like "Task".

‘:link’ Link the item headlines in the table to their origins.

‘:narrow’ An integer to limit the width of the headline column in the Org table. If you write it like ‘50!’, then the headline is also shortened in export.

‘:indent’ Indent each headline field according 従って to its level. そのレベルに

‘:tcolumns’
Number of columns to be used for times. If this is smaller than ‘:maxlevel’, lower levels are lumped into one column.

‘:level’ Should a level number column be included?

‘:sort’ A cons cell コンスセル containing 含んでいる the column to sort and a sorting type. E.g., 例えば‘:sort (1 . ?a)’ は sorts ソートします the first column alphabetically.

‘:compact’
Abbreviation for ‘:level nil :indent t :narrow 40! :tcolumns 1’. All are overwritten except if there is an explicit ‘:narrow’.

‘:timestamp’
A timestamp for the entry, when available. Look for ‘SCHEDULED’, ‘DEADLINE’, ‘TIMESTAMP’ and ‘TIMESTAMP_IA’ special properties（節 7.2 [Special Properties], ページ 68, を見てください）、in this order.

‘:tags’ When this flag このフラグが is non-nil, nil でない値のときには show 表示します the headline’s tags.

‘:properties’
List of properties shown in the table. Each property gets its own column.

‘:inherit-props’
When this flag is non-nil, nil でない値のときには the values for ‘:properties’ are inherited.

‘:formula’
Content of a ‘TBLFM’ keyword to be added and evaluated. As a special case, ‘:formula %’ adds a column with % time. If you do not specify a formula here, any existing formula below the clock table survives updates and is evaluated.

‘:formatter’
A function to format clock data and insert it into the buffer.

To get 得るには a clock summary 時間の計測のまとめを of the current level 1 tree, for the current day, you could write:

```
#+BEGIN: clocktable :maxlevel 2 :block today :scope tree1 :link t
```

[14] Language terms can be set 設定することができます through the variable 編集org-clock-clocktable-language-setup をとおして

```
#+END: clocktable
```

To use 使うには a specific time range 特定の時間の範囲を you could write このように書くことができます

8.4.3 Resolving idle time and continuous clocking

Resolving idle time

If you clock in on a work item, and then walk away from your computer---perhaps to take a phone call---you often need to "resolve" the time you were away by either subtracting it from the current clock, or applying it to another one.

By customizing カスタマイズすることによって the variable 変数`org-clock-idle-time` を to some integer, 何らかの整数に such as 10 や 15 といった Emacs can alert you when you get back to your computer after being idle for that many minutes[15], and ask what you want to do with the idle time. There will be a question waiting for you when you get back, indicating how much idle time has passed constantly updated with the current amount, as well as a set of choices to correct the discrepancy:

`k` To keep 保つ some or all of the minutes and stay clocked in, press `k`. Org asks how many of the minutes to keep. Press `RET` to keep them all, effectively changing nothing, or enter a number to keep that many minutes.

`K` If you use the shift key and press `K`, it keeps however many minutes 全ての時間を you request and then immediately clock out of that task. If you keep all of the minutes, this is the same as just clocking out of the current task.

`s` To keep 保たず none of the minutes, use `s` to subtract all the away time from the clock, and then check back in from the moment you returned.

`S` To keep 保たない none of the minutes and just clock out at the start of the away time, use the shift key and press `S`. Remember that using shift always 常に leave you clocked out, no matter which option you choose.

`C` To cancel キャンセルするには the clock altogether, 完全に時間の計測を use `C` を使ってください。Note 注意してください that if instead かわりに of canceling キャンセルする you subtract the away time, and the resulting clock amount is less than a minute, the clock is still canceled rather than cluttering up the log with an empty entry.

What if you subtracted those away minutes from the current clock, and now want to apply them to a new clock? Simply 単純に clock in to any task immediately after すぐ後に the subtraction. Org will notice that you have subtracted time "on the books", so to speak, and will ask if you want to apply those minutes to the next task you clock in on.

There is one other instance when this clock resolution magic occurs. Say you were clocked in and hacking away, and suddenly your cat chased a mouse who scared a hamster

[15] On computers using macOS, idleness is based on actual user idleness, not just Emacs' idle time. For X11, you can install a utility program '`x11idle.c`', available in the '`contrib/scripts/`' directory of the Org Git distribution, or install the xprintidle package and set it to the variable 変数`org-clock-x11idle-program-name` if you are running Debian, to get the same general treatment of idleness. On other systems, idle time refers to Emacs idle time only.

that crashed into your UPS's power button! You suddenly lose all your buffers, but thanks to auto-save you still have your recent Org-mode changes, including your last clock in.

If you restart Emacs and clock into any task, Org will notice that you have a dangling clock which was never clocked out from your last session. Using that clock's starting time as the beginning of the unaccounted-for period, Org will ask how you want to resolve that time. The logic and behavior is identical to dealing with away time due to idleness; it is just happening due to a recovery event rather than a set amount of idle time.

You can also check all the files visited by your Org agenda for dangling clocks at any time using *M-x org-resolve-clocks RET* (or *C-c C-x C-z*).

Continuous clocking

You may want to start clocking from the time 時間から when you clocked out the previous task. To enable this これを systematically, システムワイドで有効にするには、set `org-clock-continuously` を to non-`nil`. `nil` でない値に設定してください。Each time 毎回 you clock in, Org retrieves the clock-out time of the last clocked entry for this session, and start the new clock from there. そこから

If you only want this from time to time, たまにだけ use three universal prefix arguments with `org-clock-in` and two *C-u C-u* with `org-clock-in-last`.

8.5 Effort Estimates

If you want to plan your work in a very detailed way, or if you need to produce offers オファーを with quotations of the estimated work effort, you may want to assign effort estimates to entries. If you are also clocking your work, you may later 後で want to compare 比較すると良いかもしれません the planned effort with the actual working time, a great way to improve planning estimates. Effort estimates are stored in a special property '`EFFORT`'. You can set 設定することができます the effort 工数を for an entry with the following commands: 次のコマンドを使って

C-c C-x e (`org-set-effort`)
: Set the effort estimate for the current entry. With a prefix argument, 1 つの前置引数といっしょだしたお set it それを to the next allowed value 次の許可されている値に---see below. This command このんコマンドは is also accessible from the agenda with the *e* key.

C-c C-x C-e (`org-clock-modify-effort-estimate`)
: Modify the effort estimate of the item currently being clocked.

Clearly 明らかに the best way 一番良い方法は to work 作業するための with effort estimates 工数見積もりで is through column view 列ビューをとおすということです（節 7.5 [Column View], ページ 70, を見てください）。You should start はじめるえきです by setting up セットアップすることによって discrete values きちんとした値を for effort estimates, 工数見積もりに対して and a '`COLUMNS`' format that displays 表示します these values together with clock sums ---if you want to clock your time. For a specific buffer 特定のバッファに対しては、you can use: 次を使うことができます:

```
#+PROPERTY: Effort_ALL 0 0:10 0:30 1:00 2:00 3:00 4:00 5:00 6:00 7:00
#+COLUMNS: %40ITEM(Task) %17Effort(Estimated Effort){:} %CLOCKSUM
```

or, even better, you can set up セットアップすることができます these values globally by customizing カスタマイズすることによって the 変数`org-global-properties` と`org-columns-default-format` を In particular if you want to use this setup also in the agenda, a global setup may be advised.

The way to assign estimates to individual items is then to switch to column mode, and to use *S-RIGHT* and *S-LEFT* to change the value. The values you enter are immediately summed up in the hierarchy. In the column next to it, any clocked time is displayed.

If you switch 切り替えたら to column view 列ビューへ in the daily/weekly agenda, the effort column summarizes the estimated work effort for each day[16], and you can euse this これを使うことができます to find space スペースを見つけるために、in your schedule. スケジュールの中 de To get 得るには an overview 概要を of the entire part 全体部分の of the day 日の that is committed, コミットされている you can set 設定することができます the option `org-agenda-columns-add-appointments-to-effort-sum`. The appointments アポイントメントは on a day that take place over a specified time interval are then also added to the load estimate 負荷の見積もりに of the day. その日の

Effort estimates 工数見積もりを can be used 使うことができます in secondary agenda filtering 2 次的なアジェンダフィルターの中で that is triggered トリガーがかけられる with the / キーを使って in the agenda アジェンダの中で（節 10.5 [Agenda Commands], ページ 121, を見てください）。If you have these estimates これらの見積もりを defined 定義していたら consistently, two or three key presses narrow down the list to stuff that fits into an available time slot. 利用可能な時間のスロットに

8.6 Taking Notes with a Relative Timer

Org provides 提供しています two types of timers. 2 つの種類のタイマーを There is a relative せい timer 相対的タイマーがあります that counts up, カウントアップする which これが can be useful 役に立つことがあります when taking notes during, for example, 例えば a meeting or a video viewing. There is also a countdown timer.

The relative and countdown are started with separate commands.

C-c C-x 0 (`org-timer-start`)
: Start はじめるか、or reset リセットします。the relative timer. 相対的タイマーを By default, デフォルトで the timer is set 設定されます to 0. When called 呼ばれるときには with a *C-u* prefix, prompt プロンプトを出します the user ユーザーに for a starting offset. If there is a timer string at point, this is taken as the default, providing a convenient way to restart taking notes after a break in the process. When called 呼ばれるときには with a double prefix argument *C-u C-u*, change all timer strings in the active region by a certain amount. This これを使うことができます。can be used to fix 修正するために timer strings タイマー文字列を if the timer タイマーが was not started はじまっていなかったら at exactly the right moment. 正確に正しい瞬間に

C-c C-x ; (`org-timer-set-timer`)
: Start a countdown timer. The user is prompted for a duration. `org-timer-default-timer` sets the default countdown value. Giving a numeric prefix

[16] Please どうか note 注意してください the pitfalls of summing hierarchical data in a flat list（節 10.8 [Agenda Column View], ページ 135, を見てください）。

argument overrides 上書きします this default value. This command is available as `;` として in agenda buffers.

Once started, relative and countdown timers are controlled コントロールされます with the same commands. 同じコマンドを使って

C-c C-x . (`org-timer`)
: Insert 挿入します a relative time 相対的な時間を into the buffer. The first time you use this, the timer starts. Using a prefix argument restarts it.

C-c C-x - (`org-timer-item`)
: Insert 挿入します a description list item with the current relative time. With a prefix argument, 1 つの前置引数といっしょだと first 最初に reset the timer to 0. タイマーを 0 にリセットしんます。

M-RET (`org-insert-heading`)
: Once the timer list is started, you can also use *M-RET* to insert new timer items.

C-c C-x , (`org-timer-pause-or-continue`)
: Pause the timer, or continue it if it is already paused.

C-c C-x _ (`org-timer-stop`)
: Stop the timer. After this, you can only start a new timer, not continue the old one. This command also removes 取り除きます the timer from the mode line.

9 Capture, Refile, Archive

An important part of any organization system is the ability to quickly 素早く capture 収集して new ideas and tasks, and to associate reference material with them. Org does this using a process called *capture*. It also can store files related to a task (*attachments*) in a special directory. Once in the system, tasks and projects need to be moved around. Moving completed project trees to an archive file keeps the system compact and fast.

9.1 Capture

Capture 収集は lets 可能にします you quickly 素早く store 保存することを notes ノートを with little interruption of your work flow. Org's method for capturing new items is heavily inspired 強くインスピレーションを受けました by John Wiegley の素晴しい Remember パッケージに

9.1.1 Setting up capture

The following customization sets a default target file for notes.

```
(setq org-default-notes-file (concat org-directory "/notes.org"))
```

You may also define a global key for capturing new material（節 1.3 [Activation], ページ 3, を見てください）。

9.1.2 Using capture

M-x org-capture (`org-capture`)
: Display the capture templates menu. If you have templates defined（節 9.1.3 [Capture templates], ページ 93, を見てください）、it それは offers 提供します these templates これらのテンプレートを for selection or use a new Org outline node as the default template. It inserts 挿入します the template テンプレートを into the target file and switch to an indirect buffer narrowed to this new node. You may then insert the information you want.

C-c C-c (`org-capture-finalize`)
: Once you have finished entering information into the capture buffer, *C-c C-c* は returns you to the window configuration before the capture process, so that you can resume your work without further distraction. さらに気を散らすことなく When called with a prefix 1 つの前置引数といっしょ呼ばれるときには argument, finalize 収集し and then jump to the captured item.

C-c C-w (`org-capture-refile`)
: Finalize the capture process by refiling the note to a different place（節 9.5 [Refile and Copy], ページ 104, を見てください）。Please どうか realize 認識してください that this これが is a normal refiling command that will be executed---so point position at the moment you run this command is important. If you have inserted a tree with a parent and children, first move point back to the parent. Any prefix argument given to this command is passed on to the `org-refile` command.

C-c C-k (`org-capture-kill`)
: Abort the capture process and return to the previous state.

You can also call `org-capture` in a special way from the agenda, using the *k c* key combination. With this access, any timestamps inserted by the selected capture template defaults to the date at point in the agenda, rather than to the current date.

To find 見つけるには the locations 場所を of the last stored capture, use `org-capture` を使ってください with prefix commands: 前置引数とともに

C-u M-x org-capture
: Visit the target location of a capture template. You get to select the template in the usual way.

C-u C-u M-x org-capture
: Visit the last stored capture item in its buffer.

You can also jump ジャンプすることもできます to the bookmark `org-capture-last-stored`, which これは is automatically 自動的に created 作成されます unless you set 設定しないかぎり、`org-capture-bookmark` を`nil` に

To insert 挿入するには the capture 収集を at point ポイントに in an Org buffer, Org-mode のバッファの中に call 呼出してください`org-capture` を with a *C-0* 前置引数とともに

9.1.3 Capture templates

You can use 使うことができます templates テンプレートを for different types of capture items, 異なる種類の収集のアイテムと and for different target locations. The easiest way to create 作成するための such templates is through the customize interface.

C
: Customize カスタマイズします。the variable 変数`org-capture-templates` を

Before we give the formal description of template definitions, let's look at an example. Say you would like to use one template to create general TODO entries, and you want to put these entries under the heading '`Tasks`' in your file '`~/org/gtd.org`'. Also, また a date tree in the file '`journal.org`' should capture 収集すべき journal entries. 日誌のエントリーを A possible configuration would look like: このように見えます:

```
(setq org-capture-templates
      '(("t" "Todo" entry (file+headline "~/org/gtd.org" "Tasks")
         "* TODO %?\n  %i\n  %a")
        ("j" "Journal" entry (file+datetree "~/org/journal.org")
         "* %?\nEntered on %U\n  %i\n  %a")))
```

If you then 次にん press 押したら*t* を from the capture menu, Org Org-mode は will prepare 準備します the template このテンプレートを for you like this: このようにして

```
* TODO
  [[file:LINK TO WHERE YOU INITIATED CAPTURE]]
```

During expansion of the template, '`%a`' has been replaced by a link to the location from where you called the capture command. This これが can be extremely useful 極度に役に立つことがあります for deriving tasks タスクを from emails, for example. 例えば You fill in the task definition, press *C-c C-c* を押したら and Org returns 戻しまますう you あなたを to the same place 同じ場所へ where you started はじめた the capture process.

To define 定義するには special keys 特別なキーを to capture to a particular template without going through とおることなく the interactive template selection, you can create 作成することができます your key binding like this: このようにして

```
(define-key global-map "\C-cx"
  (lambda () (interactive) (org-capture nil "x")))
```

9.1.3.1 Template elements

Now ここで lets look at みましょう the elements 要素を of a template definition. テンプレート定義の Each entry as a string, characters only, for example 例えば‘`"a"`’, for a template to be selected with a single key, or ‘`"bt"`’ for selection with two keys. When using several keys, 複数のキーを使うときには、keys キーは、using 使っている the same prefix key 同じ前置キーを must be sequential 連続していなければなりません in the list and preceded by a 2-element entry explaining the prefix key, for example: 例えば

```
("b" "Templates for marking stuff to buy")
```

If you do not define a template for the `C` key, this key opens the Customize buffer for this complex variable. この複雑な変数に対する

description
: A short string describing the template, shown during selection.

type
: The type of entry, a symbol. Valid values are:

  `entry`
  : An Org-mode node, with a headline. Will be filed as the child of the target entry or as a top-level entry. The target file should be an Org file. Org-mode のファイルであるべきです

  `item`
  : A plain list item, placed in the first plain list at the target location. Again the target file should be an Org file. Org-mode のファイルであるべきです

  `checkitem`
  : A checkbox item. This only differs from the plain list item by the default template.

  `table-line`
  : A new line in the first table at the target location. Where exactly the line will be inserted depends 依存します on the properties `:prepend` and `:table-line-pos`（下を見てください））.

  `plain`
  : Text to be inserted as it is.

target
: Specification of where the captured item 収集したアイテムが should be placed. どこに置かれるべきかの In Org files, Org-mode のファイルの中で targets ターゲットは usually 通常は define 定義します a node. ノードを Entries will become children of this node. Other types will be added to the table or list in the body of this node. Most target specifications contain 含んでいます a file name. If that file name is the empty string, it defaults デフォルトで to `org-default-notes-file` になります A file can also be given as a variable or as a function called with no argument. When an absolute path is not specified for a target, it is taken とられます as relative 相対的であると to `org-directory` に対して

Valid values are:

‘`(file "path/to/file")`’
Text

will be placed at the beginning or end of that file.

‘`(id "id of existing org entry")`’
Filing as child of this entry, or in the body of the entry.

‘`(file+headline "filename" "node headline")`’
Fast configuration if the target heading is unique in the file.

‘`(file+olp "filename" "Level 1 heading" "Level 2" ...)`’
For non-unique headings, the full path is safer.

‘`(file+regexp "filename" "regexp to find location")`’
Use 使います a regular expression to position point.

‘`(file+olp+datetree "filename" [ "Level 1 heading" ...])`’
This target[1] creates 作成します a heading in a date tree[2] for today's date. If the optional outline path is given, the tree will be built under the node it is pointing to, instead かわりに of at top level. トップレベルの Check out the `:time-prompt` and `:tree-type` properties below for additional options.

‘`(file+function "filename" function-finding-location)`’
A function to find the right location in the file.

‘`(clock)`’ File to the entry that is currently being clocked.

‘`(function function-finding-location)`’
Most general way: write your own function which both visits the file and moves point to the right location.

template The template for creating the capture item. If you leave this empty, an appropriate default template will be used. Otherwise そうでなければ this これは is a string with escape codes, which これは will be replaced 置き換えられます depending on 依存して time and context 時間とコンテキストに of the capture call. 収集の呼出しの The string with escapes may be loaded from a template file, using the special syntax ‘`(file "template filename")`’. さらに詳細については下を見てください。

properties The rest of the entry is a property list of additional options. Recognized properties are:

`:prepend` Normally 通常は new captured information will be appended at the target location (last child, last table line, last list item, ...). Setting this property changes that.

[1] Org used to offer four different targets for date/week tree capture. Now, Org automatically 自動的に translates 変換します these to use `file+olp+datetree`, applying the `:time-prompt` and `:tree-type` properties. Please どうか rewrite 書き直してください your date/week-tree targets using `file+olp+datetree` since the older targets are now deprecated.

[2] A date tree is an outline structure with years on the highest level, months or ISO weeks as sublevels and then dates on the lowest level. Tags are allowed in the tree structure.

`:immediate-finish`
: When set, do not offer to edit the information, just file it away immediately. This これには makes sense 意味があります if the template only needs information 情報だけを that can be added automatically. 自動的に追加することができる

`:empty-lines`
: Set this to the number of lines to insert before and after the new item. Default 0, デフォルトは 0 であり、and the only other common value is 1.

`:clock-in`
: Start the clock in this item.

`:clock-keep`
: Keep the clock running when filing the captured entry.

`:clock-resume`
: If starting the capture interrupted a clock, restart that clock when finished with the capture. Note 注意してください that `:clock-keep` が has precedence 優先される over ˜:clock-resume˜よりも When setting 設定するときに both to 両方を non-`nil`, `nil` でない値に the current clock will run and the previous one will not be resumed.

`:time-prompt`
: Prompt for a date/time to be used for date/week trees and when filling the template. Without this property, このプロパティがなければ capture 収集は uses 使います the current date and time. Even if this property has not been set, you can force 強制することができます the same behavior by calling 呼び出すことによって`org-capture` を with a *`C-1`* 前置引数とともに

`:tree-type`
: When `week` のときには、make 作ります a week tree instead かわりに of the month tree, i.e., すなわち place the headings for each day under a heading with the current ISO week.

`:unnarrowed`
: Do not narrow the target buffer, simply 単純に show the full buffer. Default デフォルトは is to narrow 狭めるということですう it それを so that you only see the new material.

`:table-line-pos`
: Specification of the location 場所の in the table where the new line should be inserted. 挿入されるべき It should be a string 文字列であるべきです like '`II-3`' meaning that the new line should become the third line before the second horizontal separator line.

`:kill-buffer`
: If the target file was not yet visited when capture was invoked, kill the buffer again after capture is completed.

`:no-save` Do not save the target file after finishing the capture.

9.1.3.2 Template expansion

In the template itself, special "%-escapes"[3] allow 可能にします dynamic insertion of content. コンテンツの動的な挿入を The templates are expanded 展開されます in the order given here: ここで指定される順番で

'`%[FILE]`' Insert 挿入します the contents of the file given by *FILE*.

'`%(EXP)`' Evaluate Elisp expression *EXP* and replace it with the result. The *EXP* フォームは must return a string. 文字列を返さなければなりません Only placeholders pre-existing within the template, or introduced with '`%[file]`', are expanded this way. Since this これが happens after expanding non-interactive "%-escapes", those これらを can be used 使うことができます to fill the expression.

'`%<FORMAT>`'
The result of format-time-string on the *FORMAT* specification.

'`%t`' Timestamp, date only.

'`%T`' Timestamp, with date and time.

'`%u`', '`%U`' Like 上の'`%t`' と'`%T`' に似ていますうが、but inactive timestamps. 非アクティブなタイムスタンプです。

'`%i`' Initial content, the region when capture is called while the region is active. If there is text before '`%i`' on the same line, such as indentation, and '`%i`' is not inside a '`%(exp)`' form, that prefix is added before every line in the inserted text.

'`%a`' Annotation, normally 通常は the link created with `org-store-link`.

'`%A`' Like '`%a`' に似ていますが but prompt プロンプトを出します for the description part. 説明部分を求めて

'`%l`' Like '`%a`' に似ていますが、but only insert the literal link. そのままのリンクだけを挿入します。

'`%c`' Current kill ring head.

'`%x`' Content of the X clipboard.

'`%k`' Title of the currently clocked task.

'`%K`' Link to the currently clocked task.

'`%n`' User name (taken from `user-full-name`).

'`%f`' File visited by current buffer when org-capture was called.

'`%F`' Full path of the file or directory visited by current buffer.

'`%:keyword`'
Specific information for certain link types, see below.

'`%^g`' Prompt for tags, with completion on tags in target file.

[3] If you need one of these sequences literally, escape the '`%`' with a backslash.

‘%^G’ Prompt for tags, with completion all tags in all agenda files.

‘%^t’ Like ‘%t’ に似ていますが but prompt for date. 日付を求めてプロンプトを出します Similarly ‘%^T’, ‘%^u’, ‘%^U’. You may define 定義することができます a prompt like ‘%^{Birthday}t’.

‘%^C’ Interactive selection of which kill or clip to use.

‘%^L’ Like ‘%^C’ に似ていますが、but insert as link. リンクとして挿入します。

‘%^{PROP}p’
Prompt the user for a value for property *PROP*.

‘%^{PROMPT}’
Prompt the user for a string and replace this sequence with it. You may specify a default value and a completion table with ‘%^{prompt|default|completion2|completion3...}’. The arrow keys access a prompt-specific history.

‘%\N’ Insert 挿入します the text entered at the Nth ‘%^{PROMPT}’, where N is a number, starting from 1.

‘%?’ After completing the template, position point here.

For specific link types, the following keywords are defined[4]:

Link type	**Available keywords**
bbdb	‘%:name’, ‘%:company’
irc	‘%:server’, ‘%:port’, ‘%:nick’
mh, rmail	‘%:type’, ‘%:subject’, ‘%:message-id’ ‘%:from’, ‘%:fromname’, ‘%:fromaddress’ ‘%:to’, ‘%:toname’, ‘%:toaddress’ ‘%:date’ (message date header field) ‘%:date-timestamp’ (date as active timestamp) ‘%:date-timestamp-inactive’ (date as inactive timestamp) ‘%:fromto’ (either "to NAME" or "from NAME")[5]
gnus	‘%:group’, for messages also all email fields
w3, w3m	‘%:url’
info	‘%:file’, ‘%:node’
calendar	‘%:date’
org-protocol	‘%:link’, ‘%:description’, ‘%:annotation’

9.1.3.3 Templates in contexts

To control コントロールするには whether a capture template ある収集のテンプレートが should be accessible アクセス可能かどうかを from a specific context, you can customize カスタマイズすることができます。`org-capture-templates-contexts` を Let's say しましょう for example, 例えば that you have a capture template "p" for storing 保存するための

[4] If you define your own link types（節 A.3 [Adding Hyperlink Types], ページ 258, を見てください）any property you store with `org-store-link-props` can be accessed in capture templates in a similar way.

[5] This これは is always 常に the other, not the user. See the variable 変数`org-from-is-user-regexp`.

Gnus の E メールを containing 含んでいる patches. パッチを Then そうすると、you would configure 設定してください this option like this: このようにしてこのオプションを

```
(setq org-capture-templates-contexts
      '(("p" (in-mode . "message-mode"))))
```

You can also tell that the command key *p* should refer 参照するべきだ to another template. In that case, その場合には add 追加してください this command key このコマンドキーを like this: このようにして

```
(setq org-capture-templates-contexts
      '(("p" "q" (in-mode . "message-mode"))))
```

See 見てください。the docstring ドキュメント文字列を of the variable この変数の for more information. さらに情報については、

9.2 Attachments

It is often useful しばしば役に立ちます to associate reference material with an outline node/task. Small chunks of plain text can simply 単純に be stored 保存することができます in the subtree of a project. Hyperlinks ハイパーリンクが（章 4 [Hyperlinks], ページ 40, を見てください）can establish associations with files that live elsewhere on your computer or in the cloud, like emails or source code files belonging to a project. Another method is *attachments*, which これらは are files ファイルです located ある in a directory ディレクトリの中い belonging to an outline node. Org Org-mode は uses 使います directories named by the unique ID of each entry. These directories are located in the '`data/`' directory which lives in the same directory where your Org file lives[6]. If you initialize this directory with '`git init`', Org automatically 自動的に commits コミットします changes when it sees them. The attachment system has been contributed to Org by John Wiegley.

In cases where it seems better to do so, you can attach a directory of your choice to an entry. You can also make children inherit the attachment directory from a parent, so that an entire subtree uses 使います the same attached directory.

The following commands deal with attachments:

C-c C-a (`org-attach`)
: The dispatcher for commands related to the attachment system. After these keys, a list of commands is displayed and you must press 押さなければなりません an additional key 追加のキーを to select a command: コマンドを選択するためえの

  `a` (`org-attach-attach`)
  : Select a file and move it into the task's attachment directory. The file is copied, moved, or linked, depending on 依存して`org-attach-method` に Note 注意してください that hard links are not supported on all systems.

  c/*m*/*l*
  : Attach a file using the copy/move/link method. Note 注意してください that hard links are not supported on all systems.

[6] If you move entries or Org files from one directory to another, you may want to configure 設定すると良いかもしれません`org-attach-directory` を to contain 含むように an absolute path.

b (`org-attach-buffer`)
: Select a buffer and save it as a file in the task's attachment directory.

n (`org-attach-new`)
: Create 作成します a new attachment as an Emacs buffer.

z (`org-attach-sync`)
: Synchronize the current task with its attachment directory, in case you added attachments yourself.

o (`org-attach-open`)
: Open current task's attachment. If there is more than one, prompt for a file name first. Opening follows the rules set by `org-file-apps`. For more details, see the information on following hyperlinks （節 4.5 [Handling Links], ページ 43, を見てください）。

O (`org-attach-open-in-emacs`)
: Also また open 開きます the attachment, but force opening the file in Emacs.

f (`org-attach-reveal`)
: Open the current task's attachment directory.

F (`org-attach-reveal-in-emacs`)
: Also これも open the directory, but force using Dired in Emacs.

d (`org-attach-delete-one`)
: Select and delete a single attachment.

D (`org-attach-delete-all`)
: Delete all of a task's attachments. A safer way is to open the directory in Dired and delete from there.

s (`org-attach-set-directory`)
: Set a specific directory as the entry's attachment directory. This works by putting the directory path into the '`ATTACH_DIR`' property.

i (`org-attach-set-inherit`)
: Set the '`ATTACH_DIR_INHERIT`' property, so that children use the same directory for attachments as the parent does.

It is possible to attach files to a subtree from a Dired buffer. To use this feature, この機能を使うには have one window in Dired mode containing 含んでいる the file(s) to be attached and another window with point in the subtree that shall get the attachments. In the Dired window, with point on a file, *M-x org-attach-dired-to-subtree* attaches the file to the subtree using the attachment method set by variable 変数`org-attach-method` によって When files are marked in the Dired window then all marked files get attached.

Add the following lines to the Emacs init file to have *C-c C-x* `a` attach files in Dired buffers.

```
(add-hook 'dired-mode-hook
          (lambda ()
            (define-key dired-mode-map
```

```
                (kbd "C-c C-x a")
                #'org-attach-dired-to-subtree)))
```

The following code shows 示しています how to bind バインドする方法を the previous command with a specific attachment method.

```
(add-hook 'dired-mode-hook
          (lambda ()
            (define-key dired-mode-map (kbd "C-c C-x c")
              (lambda ()
                (interactive)
                (let ((org-attach-method 'cp))
                  (call-interactively #'org-attach-dired-to-subtree))))))
```

9.3 RSS Feeds

Org can add 追加することができます and change 変更することができます entries based on information 情報に found in RSS feeds and Atom feeds. You could use this to make a task out of each new podcast in a podcast feed. Or you could use a phone-based note-creating service on the web to import tasks into Org. To access アクセスするには feeds, フィードに configure 設定してください the 変数`org-feed-alist` を The docstring ドキュメント文字列に of this variable この変数の has detailed information. With the following

```
(setq org-feed-alist
      '(("Slashdot"
         "http://rss.slashdot.org/Slashdot/slashdot"
         "~/txt/org/feeds.org" "Slashdot Entries")))
```

new items from the feed provided by '`rss.slashdot.org`' result in new entries in the file '`~/org/feeds.org`' under the heading '`Slashdot Entries`', whenever いつでも the following command is used:

C-c C-x g (`org-feed-update-all`)
: Collect items from the feeds configured in `org-feed-alist` and act upon them.

C-c C-x G (`org-feed-goto-inbox`)
: Prompt for a feed name and go to the inbox configured for this feed.

Under the same headline, Org creates 作成します a ドロワー'`FEEDSTATUS`' を in which その中に it それは stores 保存する information 情報を about the status of items in the feed, to avoid adding the same item several times.

For more information, including 含む how to read 読む方法を atom feeds, see 見てください'`org-feed.el`' and the docstring of `org-feed-alist`.

9.4 Protocols for External Access

Org protocol is a means to trigger custom actions in Emacs from external applications. Any application that supports サポートしています calling external programs with an URL as argument may be used with this functionality. For example, 例えば you can configure 設定することができます bookmarks in your web browser to send a link to the current page to Org and create a note from it using capture（節 9.1 [Capture], ページ 92, を見てくだ

さい）。You can also create a bookmark that tells Emacs to open the local source file of a remote website you are browsing.

In order to use Org protocol from an application, you need to register ‘`org-protocol://`’ as a valid scheme-handler. External calls are passed to Emacs through the ‘`emacsclient`’ command, so you also need to ensure an Emacs server is running. More precisely, when the application calls

```
emacsclient org-protocol://PROTOCOL?key1=val1&key2=val2
```

Emacs calls the handler associated to *PROTOCOL* with argument ‘`(:key1 val1 :key2 val2)`’.

Org protocol comes with three predefined protocols, detailed in the following sections. Configure 設定してください`org-protocol-protocol-alist` を to define 定義するには your own. あなた自身のプロトコルを

9.4.1 `store-link` protocol

Using `store-link` handler, you can copy links, insertable through *`M-x org-insert-link`* or yanking thereafter. More precisely, the command

```
emacsclient org-protocol://store-link?url=URL&title=TITLE
```

stores the following link:

```
[[URL][TITLE]]
```

In addition, 加えて*URL* が is pushed on the kill-ring for yanking. You need to encode *URL* and *TITLE* if they contain slashes, and probably quote those for the shell.

To use 使うには this feature from a browser, add a bookmark with an arbitrary name, e.g., 例えば‘`Org: store-link`’ and enter this as *Location*:

```
javascript:location.href='org-protocol://store-link?url='+
      encodeURIComponent(location.href);
```

9.4.2 `capture` protocol

Activating `"capture"` handler pops up a ‘`Capture`’ buffer and fills the capture template associated to the ‘`X`’ key with them.

```
emacsclient org-protocol://capture?template=X?url=URL?title=TITLE?body=BODY
```

To use 使うには this feature, add a bookmark with an arbitrary name, e.g., 例えば‘`Org: capture`’, and enter this as ‘`Location`’:

```
javascript:location.href='org-protocol://capture?template=x'+
      '&url='+encodeURIComponent(window.location.href)+
      '&title='+encodeURIComponent(document.title)+
      '&body='+encodeURIComponent(window.getSelection());
```

The result depends on the capture template used, which これは is set 設定されています in the bookmark itself, as in the example above, or in `org-protocol-default-template-key`.

The following template placeholders are available:

```
%:link          The URL
%:description   The webpage title
%:annotation    Equivalent to [[%:link][%:description]]
%i              The selected text
```

9.4.3 `open-source` protocol

The `open-source` handler is designed to help with editing local sources when reading a document. To that effect, そのん効果のためには、you can use 使うことができます a bookmark ブックマークを with the following location: 次のロケーションを持つ

```
javascript:location.href='org-protocol://open-source?&url='+
      encodeURIComponent(location.href)
```

The variable 変数`org-protocol-project-alist` は maps マップします URLs to local file names, by stripping URL parameters from the end and replacing the `:base-url` with `:working-directory` and `:online-suffix` with `:working-suffix`. For example, 例えば assuming 仮定して you own a local copy of '`https://orgmode.org/worg/`' contents at '`/home/user/worg`', you can set `org-protocol-project-alist` to the following

```
(setq org-protocol-project-alist
      '(("Worg"
         :base-url "https://orgmode.org/worg/"
         :working-directory "/home/user/worg/"
         :online-suffix ".html"
         :working-suffix ".org")))
```

If you are now browsing '`https://orgmode.org/worg/org-contrib/org-protocol.html`' and find a typo or have an idea about how to enhance the documentation, simply 単純に click the bookmark and start editing.

However, しかし such mapping may not yield the desired results. Suppose しましょう you maintain an online store located ある at '`http://example.com/`'. The local sources reside in '`/home/user/example/`'. It is common practice to serve all products in such a store through one file and rewrite URLs that do not match an existing file on the server. That way, a request to '`http://example.com/print/posters.html`' might be rewritten on the server to something like '`http://example.com/shop/products.php/posters.html.php`'. The `open-source` handler probably cannot find a file named '`/home/user/example/print/posters.html.php`' and fails.

Such an entry in `org-protocol-project-alist` may hold an additional property `:rewrites`. This property is a list of cons cells, each of which maps a regular expression to a path relative 相対的な to the `:working-directory`.

Now map マップします the この URL を to the path '`/home/user/example/products.php`' へ by adding 追加することによって`:rewrites` のルールを like this: このようにして

```
(setq org-protocol-project-alist
      '(("example.com"
         :base-url "http://example.com/"
         :working-directory "/home/user/example/"
         :online-suffix ".php"
         :working-suffix ".php"
         :rewrites (("example.com/print/" . "products.php")
                    ("example.com/$" . "index.php")))))
```

Since '`example.com/$`' is used as a regular expression, it maps '`http://example.com/`', '`https://example.com`', '`http://www.example.com/`' and similar to '`/home/user/example/index.php`'.

The `:rewrites` rules are searched as a last resort if and only if no existing file name is matched.

Two functions can help you filling `org-protocol-project-alist` with valid contents: `org-protocol-create` and `org-protocol-create-for-org`. The latter is of use if you're editing an Org file that is part of a publishing project.

9.5 Refile and Copy

When reviewing レビューするときには the captured data, 収集したデータを you may want to refile or to copy some of the entries into a different list, for example 例えば into a project. Cutting, finding the right location, and then pasting the note is cumbersome. やっかいです To simplify this process, このプロセスを単純にするために、you can use the following special command: 次の特別なコマンドを使うことができます:

C-c M-w (`org-copy`)
: Copying works like refiling, except that the original note is not deleted.

C-c C-w (`org-refile`)
: Refile the entry or region at point. This command offers 提供します possible locations for refiling the entry こ and lets 可能にします you select one with completion. 補完を使って The item (or all items in the region) is filed below the target heading as a subitem. Depending on 依存して`org-reverse-note-order` に it is either the first or last subitem.

  By default, デフォルトで all level 1 headlines in the current buffer are considered to be targets, but you can have more complex definitions across a number of files. 詳細については、変数`org-refile-targets` を見てください。If you would like to select a location via a file-path-like completion along the outline path, see 見てください the variables 変数`org-refile-use-outline-path` と`org-outline-path-complete-in-steps` を見てください If you would like to be able to create new nodes as new parents for refiling on the fly, check チェックしてください the variable 変数`org-refile-allow-creating-parent-nodes`. When the variable 変数`org-log-refile`[7] が is set, 設定されているときには a timestamp or タイムスタンプか a note ノートが is recorded 記録するえます whenever いつでも an entry あるエントリーが is refiled. 再ファイルされるときには

C-u C-c C-w
: Use 使います the refile interface to jump to a heading.

C-u C-u C-c C-w (`org-refile-goto-last-stored`)
: Jump ジャンプします to the location 場所へ where `org-refile` が last 最後に moved 移動した a tree to. ツリーを

C-2 C-c C-w
: Refile as the child of the item currently being clocked.

C-3 C-c C-w
: Refile and keep the entry in place. Also see `org-refile-keep` も見てください to make this これを the default behavior, and beware that this may result in duplicated '`ID`' properties.

[7] Note 注意してください the corresponding '`STARTUP`' options '`logrefile`', '`lognoterefile`', and '`nologrefile`'.

C-0 C-c C-w or *C-u C-u C-u C-c C-w* (`org-refile-cache-clear`)
: Clear the target cache. Caching of refile targets can be turned on by setting 設定することによって`org-refile-use-cache` を To make the command このコマンドに see new possible targets, you have to clear the cache with this command.

9.6 Archiving

When a project represented by a (sub)tree is finished, you may want to move the tree out of the way and to stop it from contributing to the agenda. Archiving is important to keep your working files compact and global searches like the construction of agenda views fast.

C-c C-x C-a (`org-archive-subtree-default`)
: Archive アーカイブします the current entry using 使って the command コマンドを specified 指定されている in the variable 変数`org-archive-default-command` の中で

9.6.1 Moving a tree to an archive file

The most common archiving action is to move a project tree to another file, the archive file.

C-c C-x C-s or short *C-c $* (`org-archive-subtree`)
: Archive アーカイブします the subtree サブツリーを starting はじまる at point position ポイントの場所で to the location 場所へ given by `org-archive-location` によって指定される

C-u C-c C-x C-s
: Check if any direct children of the current headline could be moved to the archive. To do this, これを行なうためいん check each subtree for open TODO entries. If none is found, 何も見つからなかったら the command このコマンドは offers 提供します to move 移動する機会を it それを to the archive location. If point ポイントが is *not* on a headline when this command is invoked, check level 1 trees.

C-u C-u C-c C-x C-s
: As above, but check subtree for timestamps instead かわりに of TODO entries. The command このコマンドは offers 提供します to archive アーカイブする方法を the subtree if it *does* contain a timestamp, and that timestamp is in the past.

The default archive location デフォルト値のアーカイブの場所は is a file ファイルです in the same directory as the current file, with the name derived by appending '`_archive`' to the current file name. You can also choose what heading to file archived items under, with the possibility to add them to a datetree in a file. For information and examples on how to specify the file and the heading, see the documentation string of the variable 変数`org-archive-location` の

There is also an in-buffer option for setting this variable, for example:

```
#+ARCHIVE: %s_done::
```

If you would like to have a special archive location for a single entry or a (sub)tree, give the entry an '`ARCHIVE`' property with the location as the value（章 7 [Properties and Columns], ページ 66, を見てください）。

When a subtree is moved, it receives a number of special properties that record context information like the file from where the entry came, its outline path the archiving time etc. Configure 設定してください the variable 変数`org-archive-save-context-info` を to adjust 調節するには the amount 量を of information added.

9.6.2 Internal archiving

If you want to just switch off ---for agenda views ---certain subtrees without moving 移動することなく them to a different file, you can use the '`ARCHIVE`' タグを使うことができます。

A headline that is marked with the '`ARCHIVE`' tag（章 6 [Tags], ページ 62, を見てください）stays at its location in the outline tree, but behaves in the following way:

- It does not open when you attempt to do so with a visibility cycling command（節 2.3 [Visibility Cycling], ページ 7, を見てください）。You can force 強制することができまう cycling archived subtrees with *`C-TAB`* を使ってか or by setting 設定することによって the option オプション`org-cycle-open-archived-trees` を Also また normal outline commands, like `outline-show-all`, open archived subtrees.
- During sparse tree construction（節 2.6 [Sparse Trees], ページ 12, を見てください）、matches in archived subtrees are not exposed, unless you configure 設定しんあいかぎり the option オプション`org-sparse-tree-open-archived-trees` を
- During agenda view construction（章 10 [Agenda Views], ページ 108, を見てください）、the content of archived trees is ignored unless you configure 設定しないかぎり the option オプション`org-agenda-skip-archived-trees` を in which case these trees are always 常に included. In the agenda you can press `v a` to get archives temporarily included.
- Archived trees are not exported（章 12 [Exporting], ページ 145, を見てください）、only the headline is. エクスポートされます Configure 設定してください the details 詳細を using the variable 変数`org-export-with-archived-trees` を使って
- Archived trees are excluded from column view unless the variable 変数`org-columns-skip-archived-trees` が is configured to `nil`.

The following commands help manage the '`ARCHIVE`' tag:

`C-c C-x a` (`org-toggle-archive-tag`)
: Toggle トグルします the archive tag for the current headline. When the tag is set, the headline changes to a shadowed face, and the subtree below it is hidden.

`C-u C-c C-x a`
: Check if any direct children of the current headline should be archived. To do this, これを行なうために check each subtree for open TODO entries. If none is found, the command offers 提供します to set the '`ARCHIVE`' tag for the child. If point ポイントが is *not* on a headline when this command is invoked, check the level 1 trees.

`C-TAB` (`org-force-cycle-archived`)
: Cycle a tree even if it is tagged with '`ARCHIVE`'.

`C-c C-x A` (`org-archive-to-archive-sibling`)
: Move 移動します the current entry to the *Archive Sibling*. This is これは a sibling 兄弟です of the entry with the heading '`Archive`' and the archive tag.

The entry becomes a child of that sibling and in this way retains a lot of its original context, including 含む inherited tags and approximate position in the outline.

10 Agenda Views

Due ために to the way ホウホウンお Org works, TODO items, time-stamped items, and tagged headlines can be scattered throughout a file or even a number of files. To get an overview of open action items, or of events that are important for a particular date, this information must be collected, 集められ、sorted and displayed in an organized way.

Org can select items based on various criteria and display them in a separate buffer. Seven different view types are provided:

- an *agenda* that is like a calendar and shows 表示する information 情報を for specific dates,
- a *TODO list* that covers all unfinished action items,
- a *match view*, showings headlines based on the tags, properties, and TODO state associated with them,
- a *text search view* that shows 表示する all entries from multiple files that contain specified keywords,
- a *stuck projects view* showing projects that currently do not move along, and
- *custom views* that are special searches and combinations of different views.

The extracted information is displayed in a special *agenda buffer*. This buffer is read-only, このバッファは読み込み専用ですが but provides 提供しています commands コマンドを to visit the corresponding locations in the original Org files, and even to edit these files remotely.

By default, デフォルトで the report ignores commented（節 12.6 [Comment Lines], ページ 153, を見てください）and archived（節 9.6.2 [Internal archiving], ページ 106, を見てください）entries. You can override 上書きすることができまますう this これを by setting 設定することによって`org-agenda-skip-comment-trees` と`org-agenda-skip-archived-trees` を to `nil` に

Two variables control how the agenda buffer is displayed and whether the window configuration is restored when the agenda exits: `org-agenda-window-setup` と`org-agenda-restore-windows-after-quit` です。

10.1 Agenda Files

The information to be shown is normally 通常は collected from all *agenda files*, the files listed in the variable 変数`org-agenda-files` の中で

10.2 The Agenda Dispatcher

The views are created through a dispatcher, accessible with *`M-x org-agenda`*, or, better, bound to a global key（節 1.3 [Activation], ページ 3, を見てください）。It displays 表示します a menu メニューを from which an additional letter is required to execute a command. The dispatcher このディスパッチャは offers 提供します the following default commands: 次のデフォルトのコマンドを

`a` Create 作成します the calendar-like agenda（節 10.3.1 [Weekly/daily agenda], ページ 110, を見てください）。

`t` or `T` Create 作成します a list of all TODO items（節 10.3.2 [Global TODO list], ページ 112, を見てください）。

`m` or `M` Create 作成します a list of headlines 見出しのリストを matching a given expression（節 10.3.3 [Matching tags and properties], ページ 113, を見てください）。

`s` Create 作成します a list of entries selected by a boolean expression of keywords and/or regular expressions that must or must not occur in the entry.

`/` Search for a regular expression in all agenda files and additionally in the files listed in `org-agenda-text-search-extra-files`. This これは uses 使います the Emacs command `multi-occur`. A prefix argument 前置引数を使うことができます。can be used to specify 指定するために the number of context lines コンテキストの行数を for each match, 角マッチにん対する default is デフォルトは 1 です。

`#` or `!` Create 作成します a list of stuck projects スタックしているプロジェクトの（節 10.3.5 [Stuck projects], ページ 116, を見てください）。

`<` Restrict an agenda command to the current buffer[1]. After pressing `<`, you still need to press the character selecting the command.

`< <` If there is an active region, restrict the following agenda command to the region. Otherwise, そうでなければ restrict 制限します it それを to the current subtree[2]. After pressing `< <`, you still need to press the character selecting the command.

`*` Toggle トグルします sticky agenda views. スティッキーアジェンダビューをトグルしま。By default, デフォルトで Org maintains only a single agenda buffer and rebuilds it each time you change the view, to make sure everything is always 常に up to date. If you switch between views often and the build time bothers you, you can turn on sticky agenda buffers (make this the default by customizing カスタマイズすることによって the variable 変数`org-agenda-sticky`) With sticky agendas, スティッキーアジェンダを使うと、the dispatcher only switches to the selected view, you need to update it by hand with `r` or `g`. You can toggle sticky agenda view any time with `org-toggle-sticky-agenda`.

You can also define custom commands that are accessible through the dispatcher, just like the default commands. This includes the possibility to create extended agenda buffers that contain several blocks together, for example 例えば the weekly agenda, the global TODO list and a number of special tags matches. 節 10.6 [Custom Agenda Views], ページ 130, を見てください。

10.3 The Built-in Agenda Views

In this section we describe the built-in views.

---

[1] For backward compatibility, you can also press `1` to restrict to the current buffer.

[2] For backward compatibility, you can also press `0` to restrict to the current region/subtree.

10.3.1 Weekly/daily agenda

The purpose of the weekly/daily *agenda* is to act like a page of a paper agenda, showing all the tasks for the current week or day.

M-x org-agenda `a` (`org-agenda-list`)
: Compile an agenda for the current week from a list of Org files. The agenda shows 表示します the entries for each day. With a numeric prefix argument[3] ---like *C-u 2 1 M-x org-agenda* `a`---you may set the number of days to be displayed.

The default number of days デフォルトの日数 displayed 表示される in the agenda アジェンダの中で is set 設定されます by the variable 変数`org-agenda-span` によって This variable この変数を can be set 設定することができます to any number of days 任意の日数に you want to see 見たい by default デフォルトで in the agenda, or to a span name, such a `day`, `week`, `month` or `year`. For weekly agendas, the default is to start on the previous Monday（~org-agenda-start-on-weekday~ を見てください）。You can also set the start date using a date shift: '`(setq org-agenda-start-day "+10d")`' starts the agenda ten days from today in the future.

Remote editing from the agenda buffer means, 意味します for example, 例えば that you can change the dates of deadlines and appointments from the agenda buffer. The commands コマンドは available in the Agenda buffer are listed in 節 10.5 [Agenda Commands], ページ 121.

Calendar/Diary integration

Emacs は contains 含んでいます the calendar and diary by Edward M. Reingold. The calendar カレンダーは displays 表示します a three-month calendar with holidays from different countries and cultures. The diary allows 可能にします you to keep track of 追うことを anniversaries, lunar phases, sunrise/set, recurrent appointments (weekly, monthly) and more. In this way, この方法で it is quite complementary to Org. It can be very useful とても役に立つことがあります to combine output from Org with the diary.

In order to include entries from the Emacs diary into Org-mode's agenda, you only need to customize カスタマイズする必用がありまう the variable この変数を

```
(setq org-agenda-include-diary t)
```

After that, その後では everything happens automatically. 自動的に起きます All diary entries including 含む holidays, anniversaries, etc., are included in the agenda buffer created by Org-mode. `SPC` と`TAB` と`RET` を can be used 使うことができます from the agenda buffer アジェンダバッファから to jump ジャンプするために to the diary ファイルへ in order to edit 編集するために existing diary entries. The *i* command to insert new entries for the current date works in the agenda buffer, as well as the commands *S*, *M*, and *C* to display 表示するため Sunrise/Sunset times, show lunar phases and to convert to other calendars, respectively. それぞれ*c* を can be used 使うことができます to switch 行ったり来たりするために back and forth between calendar and agenda. カレンダーとアジェンダの間を

[3] For backward compatibility, the universal prefix argument *C-u* causes all TODO entries to be listed before the agenda. This feature is deprecated, use the dedicated TODO list, or a block agenda instead かわりに（節 10.6.2 [Block agenda], ページ 131, を見てください）

If you are using the diary only for S-exp entries and holidays, it is faster to not use the above setting, but instead かわりに to copy or even move the entries into an Org file. Org-mode evaluates diary-style sexp entries, and does it faster because there is no overhead for first creating the diary display. Note 注意してください that the sexp entries must start at the left margin, no whitespace is allowed before them, as seen in the following segment of an Org file:[4]

```
* Holidays
  :PROPERTIES:
  :CATEGORY: Holiday
  :END:
%%(org-calendar-holiday)   ; special function for holiday names

* Birthdays
  :PROPERTIES:
  :CATEGORY: Ann
  :END:
%%(org-anniversary 1956  5 14) Arthur Dent is %d years old
%%(org-anniversary 1869 10  2) Mahatma Gandhi would be %d years old
```

Anniversaries from BBDB

If you are using the Insidious Big Brother Database to store your contacts, you very likely prefer to store anniversaries in BBDB rather than in a separate Org or diary file. Org supports サポートしています this これを and can show BBDB anniversaries as part of the agenda. All you need to do is to add the following to one of your agenda files:

```
* Anniversaries
  :PROPERTIES:
  :CATEGORY: Anniv
  :END:
%%(org-bbdb-anniversaries)
```

You can then go ahead and define anniversaries for a BBDB record. Basically, you need a field named '`anniversary`' for the BBDB record which contains the date in the format '`YYYY-MM-DD`' or '`MM-DD`', followed by a space and the class of the anniversary ('`birthday`', '`wedding`', or a format string). If you omit the class, it defaults to '`birthday`'. Here これらは are a few examples, the header for the file '`org-bbdb.el`' contains more detailed information.

```
1973-06-22
06-22
1955-08-02 wedding
2008-04-14 %s released version 6.01 of Org-mode, %d years ago
```

After a change to BBDB, or for the first agenda display during an Emacs session, the agenda display アジェンダの表示は suffers a short delay as Org updates its hash with anniversaries. However, しかし from then on それ以降では things will be very fast, much faster in fact than a long list of '`%%(diary-anniversary)`' entries in an Org or Diary file.

[4] The variable 変数`org-anniversary` used in the example is just like `diary-anniversary`, but the argument order is always 常に according 従います to ISO に and therefore よって independent of the value of `calendar-date-style`.

If you would like to see 見たいのであれば upcoming anniversaries やってくる記念日を with a bit of forewarning, 少し前もって you can use 使うことができます: the following instead: かわりに次のものを

```
* Anniversaries
  :PROPERTIES:
  :CATEGORY: Anniv
  :END:
%%(org-bbdb-anniversaries-future 3)
```

That will give you three days' warning: on the anniversary date itself and the two days prior. The argument is optional: if omitted, it defaults to 7.

Appointment reminders

Org can interact with Emacs appointments notification facility. To add 追加するには the appointments of your agenda files, use the command `org-agenda-to-appt`. This command このコマンドは lets 可能にします you filter フィルターすることを through the list of your appointments and add 追加するこていお only those belonging to a specific category or matching a regular expression. It also reads a '`APPT_WARNTIME`' property which overrides 上書きします the value of 値を`appt-message-warning-time` for this appointment. 詳細についてはドキュメント文字列を見てください。

10.3.2 The global TODO list

The global TODO list contains all unfinished TODO items formatted and collected into a single place.

M-x org-agenda t (`org-todo-list`)

Show the global TODO list. This collects the TODO items from all agenda files（章 10 [Agenda Views], ページ 108, を見てください）into a single buffer. 1 つのバッファの中へ By default, デフォルトで this lists リストします items with a state the is not a DONE state. The buffer is in `agenda-mode`, so there are commands to examine and manipulate the TODO entries directly from that buffer（節 10.5 [Agenda Commands], ページ 121, を見てください）。

M-x org-agenda T (`org-todo-list`)

Like the above, 上に似ていますが but allows selection of a specific TODO keyword. You can also do 行なうこともできます this by specifying 指定することによって a prefix argument 前置引数を to *t* に対して You are prompted プロンプトが出ます for a keyword, キーワードを求めて and you may also specify 指定することもできまう several keywords by separating them with '`|`' as the boolean OR operator. With a numeric prefix, the Nth keyword in `org-todo-keywords` is selected.

The *r* key in the agenda buffer regenerates it, and you can give a prefix argument to this command to change the selected TODO keyword, for example 例えば*3 r*. If you often need a search for a specific keyword, define a custom command for it（節 10.2 [Agenda Dispatcher], ページ 108, を見てください）。

Matching specific TODO keywords can also be done as part of a tags search（節 6.4 [Tag Searches], ページ 64, を見てください）。

Remote editing of TODO items means that you can change 変更することができます the state 状態を of a TODO entry with a single key press. The commands available in the TODO list are described in 節 10.5 [Agenda Commands], ページ 121.

Normally 通常は the global TODO list simply 単純に shows 表示します all headlines 全ての見出しを with TODO keywords. This list can become very long. このリストがとても長くなることがあります。There are two ways to keep it more compact:

- Some people view a TODO item that has been *scheduled* for execution or have a *deadline*（節 8.1 [Timestamps], ページ 75, を見てください）as no longer *open*. Configure 設定してください the variables 変数`org-agenda-todo-ignore-scheduled`, `org-agenda-todo-ignore-deadlines`, `org-agenda-todo-ignore-timestamp` and/or `org-agenda-todo-ignore-with-date` to exclude such items from the global TODO list.
- TODO items may have sublevels to break up the task into subtasks. In such cases it may be enough to list only the highest level TODO headline and omit the sublevels from the global list. Configure 設定してください the variable 変数`org-agenda-todo-list-sublevels` を to get this behavior. このふるまいを得るには、

10.3.3 Matching tags and properties

If headlines in the agenda files are marked with *tags*（章 6 [Tags], ページ 62, を見てください）、or have properties（章 7 [Properties and Columns], ページ 66, を見てください）you can select headlines based on this metadata and collect them into an agenda buffer. The match syntax described here also applies when creating sparse trees with *C-c / m*.

M-x org-agenda m (`org-tags-view`)
: Produce a list of all headlines that match a given set of tags. The command prompts プロンプトを出します for a selection criterion, which これは is a boolean logic expression 論理式です with tags, like '`+work+urgent-withboss`' or '`work|home`'（章 6 [Tags], ページ 62, を見てください）。If you often need a specific search, define a custom command for it（節 10.2 [Agenda Dispatcher], ページ 108, を見てください）。

M-x org-agenda M (`org-tags-view`)
: Like *m* に似ていますが but only select headlines that are also TODO items and force checking subitems（変数`org-tags-match-list-sublevels` を見てください）。To exclude 除外するには scheduled/deadline items, see the variable 変数`org-agenda-tags-todo-honor-ignore-options`. Matching specific TODO keywords together with a tags match is also possible, see 節 6.4 [Tag Searches], ページ 64.

The commands available in the tags list are described in 節 10.5 [Agenda Commands], ページ 121.

A search string 検索文字列は can use 使うことができます Boolean operators 論理演算子'`&`' と for AND を意味する and '`|`' を for OR を意味する'`&`' binds more strongly than '`|`' よりも強く結合します。Parentheses are currently not implemented. Each element in the search is either a tag, a regular expression matching tags, or an expression like '`PROPERTY OPERATOR VALUE`' with a comparison operator, accessing a property value. Each element may be preceded by '`-`' to select against it, and '`+`' is syntactic sugar for positive selection.

The AND operator '`&`' is optional when '`+`' or '`-`' is present. Here これらは are some examples, using only tags.

`+work-boss`
: Select headlines tagged '`work`', but discard those also tagged '`boss`'.

`work|laptop`
: Selects lines tagged '`work`' or '`laptop`'.

`work|laptop+night`
: Like before, 前に似ていますが but require the '`laptop`' lines to be tagged also '`night`'.

Instead かわりに of a tag, タグの you may also specify 指定することもできます a regular expression 正規表現を enclosed in curly braces. 中括弧の中に囲まれた For example, 例えば'`work+{^boss.*}`' は matches headlines that contain the tag '`:work:`' and any tag *starting* with '`boss`'.

Group tags（節 6.3 [Tag Hierarchy], ページ 63, を見てください）are expanded as regular expressions. E.g., 例えば if '`work`' is a group tag for the group '`:work:lab:conf:`', then searching for '`work`' also searches for '`{\(?:work\|lab\|conf\)}`' and searching for '`-work`' searches for all headlines but those with one of the tags in the group (i.e., すなわち'`-{\(?:work\|lab\|conf\)}`').

You may also test for properties（章 7 [Properties and Columns], ページ 66, を見てください）at the same time as matching tags. The properties may be real properties, or special properties that represent other metadata（節 7.2 [Special Properties], ページ 68, を見てください）。For example, 例えば the property '`TODO`' represents the TODO keyword of the entry. Or, the property '`LEVEL`' represents the level of an entry. So searching '`+LEVEL=3+boss-TODO="DONE"`' lists all level three headlines that have the tag '`boss`' and are *not* marked with the TODO keyword '`DONE`'. In buffers with `org-odd-levels-only` set, '`LEVEL`' does not count the number of stars, but '`LEVEL=2`' corresponds to 3 stars etc.

Here are more examples: これらはさらなる例です:

'`work+TODO="WAITING"`'
: Select '`work`'-tagged TODO lines with the specific TODO keyword '`WAITING`'.

'`work+TODO="WAITING"|home+TODO="WAITING"`'
: Waiting tasks both at work and at home.

When matching properties, プロパティにマッチするときには a number of different operators たくさんの異なる演算子を can be used 使うことができます to test the value of a property. Here これは is a complex example: 複雑な例です:

```
+work-boss+PRIORITY="A"+Coffee="unlimited"+Effort<2
        +With={Sarah|Denny}+SCHEDULED>="<2008-10-11>"
```

The type of comparison depends on how the comparison value is written:

- If the comparison value is a plain number, a numerical comparison is done, and the allowed operators are '`<`', '`=`', '`>`', '`<=`', '`>=`', and '`<>`'.
- If the comparison value is enclosed in double-quotes, a string comparison is done, and the same operators are allowed.

- If the comparison value is enclosed in double-quotes *and* angular brackets (like '`DEADLINE<="<2008-12-24 18:30>"`'), both values are assumed to be date/time specifications in the standard Org way, and the comparison is done accordingly. Valid values also include '`"<now>"`' for now (including 含む time), 時間を'`"<today>"`', and '`"<tomorrow>"`' for these days at 0:00 hours, i.e., すなわち without a time specification. You can also use 使うことおできます strings like '`"<+5d>"`' or '`"<-2m>"`' with units '`d`', '`w`', '`m`', and '`y`' for day, week, month, and year, respectively.
- If the comparison value is enclosed in curly braces, a regexp match is performed, with '`=`' meaning that the regexp matches the property value, and '`<>`' meaning that it does not match.

So the search string in the example finds entries tagged '`work`' but not '`boss`', which これは also have a priority value '`A`', a '`Coffee`' property with the value '`unlimited`', an '`EFFORT`' property that is numerically smaller than 2, a '`With`' property that is matched by the regular expression '`Sarah|Denny`', and that are scheduled on or after October 11, 2008.

You can configure 設定することができあす Org-mode を to use property inheritance during a search, but beware that this can slow down searches considerably. 詳細については節 7.4 [Property Inheritance], ページ 69, を見てください。

For backward compatibility, and also for typing speed, there is also a different way to test TODO states in a search. For this, terminate the tags/property part of the search string (which may include several terms connected with '`|`') with a '`/`' and then specify a Boolean expression just for TODO keywords. The syntax is then similar to that for tags, but should be applied with care: for example, 例えば a positive selection on several TODO keywords cannot meaningfully be combined with boolean AND. However, しかし*negative selection* combined with AND can be meaningful. To make sure 確実にするには that only lines are checked that actually have any TODO keyword (resulting in a speed-up), use *`M-x org-agenda M`*, or equivalently start the TODO part after the slash with '`!`'. Using *`M-x org-agenda M`* or '`/!`' does not match TODO keywords in a DONE state. Examples:

'`work/WAITING`'
: Same as '`work+TODO="WAITING"`'.

'`work/!-WAITING-NEXT`'
: Select '`work`'-tagged TODO lines that are neither '`WAITING`' nor '`NEXT`'.

'`work/!+WAITING|+NEXT`'
: Select '`work`'-tagged TODO lines that are either '`WAITING`' or '`NEXT`'.

10.3.4 Search view

This agenda view is a general text search facility for Org-mode entries. It is particularly useful 特に役に立ちます to find notes.

`M-x org-agenda s` (`org-search-view`)
: This これは is a special search that lets 可能にする you select entries by matching a substring or specific words using a boolean logic.

For example, 例えば the search string '`computer equipment`' matches entries that contain '`computer equipment`' as a substring, even if the two words are separated by more space or a line break.

Search view can also search for specific keywords in the entry, using Boolean logic. The search string '`+computer +wifi -ethernet -{8\.11[bg]}`' matches note entries that contain the keywords '`computer`' and '`wifi`', but not the keyword '`ethernet`', and which are also not matched by the regular expression '`8\.11[bg]`', meaning to exclude both '`8.11b`' and '`8.11g`'. The first '`+`' is necessary to turn on boolean search, other '`+`' characters are optional. For more details, see the docstring of the command `org-search-view`.

You can incrementally adjust a boolean search with the following keys

[Add a positive search word
] Add a negative search word
{ Add a positive regular expression
} Add a negative regular expression

Note 注意してください that in addition 加えて to the agenda files, this command also searches the files listed リストされているファイル毎も in `org-agenda-text-search-extra-files` の中で

10.3.5 Stuck projects

If you are following a system like David Allen's GTD to organize your work, one of the "duties" you have is a regular review to make sure that all projects move along. A *stuck* project is a project that has no defined next actions, so it never 決してい shows up 出現しません in the TODO lists Org mode produces. During the review, you need to identify such projects and define next actions for them.

M-x org-agenda # (`org-agenda-list-stuck-projects`)
List projects that are stuck.

M-x org-agenda !
Customize カスタマイズしまう the variable 変数`org-stuck-projects` を to define what a stuck project is and how to find it.

You almost certainly ほとんど確実に need to configure 設定する必用があふるいます this view このビューを before it works for you. The built-in default assumes that all your projects are level-2 headlines, and that a project is not stuck if it has at least one entry marked with a TODO keyword '`TODO`' or '`NEXT`' or '`NEXTACTION`'.

Let's assume that you, in your own way of using Org-mode, identify projects with a tag '`:PROJECT:`', and that you use a TODO keyword '`MAYBE`' to indicate a project that should not be considered yet. Let's further assume that the TODO keyword '`DONE`' marks finished projects, and that '`NEXT`' and '`TODO`' indicate next actions. The tag '`:@shop:`' indicates 示し shopping ショッピングを and is a next action even without the NEXT tag. Finally, 最後に if the project contains the special word '`IGNORE`' anywhere, it should not be listed either. In this case この場合には you would start はじめます by identifying 特定することによって eligible projects with a tags/TODO match（節 6.4 [Tag Searches], ページ 64, を見てください）'`+PROJECT/-MAYBE-DONE`', and then check for '`TODO`', '`NEXT`', '`@shop`', and '`IGNORE`' in the subtree to identify projects that are not stuck. The correct customization for this is:

```
(setq org-stuck-projects
      '("+PROJECT/-MAYBE-DONE" ("NEXT" "TODO") ("@shop")
        "\\<IGNORE\\>"))
```

Note 注意してください that if a project is identified as non-stuck, the subtree of this entry is searched for stuck projects.

10.4 Presentation and Sorting

Before displaying items in an agenda view, Org-mode visually prepares the items and sorts them. Each item occupies a single line. The line starts with a *prefix* that contains the *category*（節 10.4.1 [Categories], ページ 117, を見てください）of the item and other important information. You can customize カスタマイズすることができます in which column tags are displayed 表示されるのか through `org-agenda-tags-column` をとおして You can also customize カスタマイズすることができます the prefix using 使って the option `org-agenda-prefix-format`. This prefix is followed by a cleaned-up version of the outline headline associated with the item.

10.4.1 Categories

The category is a broad label assigned to each agenda item. By default, the category is simply 単純に derived 取られます from the file name, but you can also specify 指定することもできます it それを with a special line in the buffer, like this:

```
#+CATEGORY: Thesis
```

If you would like to have a special category for a single entry or a (sub)tree, give the entry a '`CATEGORY`' property with the special category you want to apply as the value.

The display in the agenda buffer looks best if the category is not longer than 10 characters.

You can set up セットアップすることができます icons for category カテゴリーに対するアイコンを by customizing カスタマイズすることによって the `org-agenda-category-icon-alist` 変数を

10.4.2 Time-of-day specifications

Org-mode checks each agenda item for a time-of-day specification. The time can be part of the timestamp that triggered inclusion into the agenda, for example 例えば

```
<2005-05-10 Tue 19:00>
```

Time ranges can be specified with two timestamps:

```
<2005-05-10 Tue 20:30>--<2005-05-10 Tue 22:15>
```

In the headline of the entry itself, a time(range)---like '`12:45`' or a '`8:30-1pm`'---may also appear as plain text[5].

If the agenda integrates the Emacs diary（節 10.3.1 [Weekly/daily agenda], ページ 110, を見てください）、time specifications in diary entries are recognized as well.

For agenda display, Org-mode は extracts the time and displays 表示します it それを in a standard 24 hour format as part of the prefix. The example times in the previous paragraphs would end up in the agenda like this: このような

```
 8:30-13:00 Arthur Dent lies in front of the bulldozer
12:45...... Ford Prefect arrives and takes Arthur to the pub
19:00...... The Vogon reads his poem
20:30-22:15 Marvin escorts the Hitchhikers to the bridge
```

[5] You can, however, disable this by setting 設定することによって`org-agenda-search-headline-for-time` 変数を to a `nil` value.

If the agenda is in single-day mode, or for the display of today, the timed entries are embedded in a time grid, like

```
 8:00...... ------------------
 8:30-13:00 Arthur Dent lies in front of the bulldozer
10:00...... ------------------
12:00...... ------------------
12:45...... Ford Prefect arrives and takes Arthur to the pub
14:00...... ------------------
16:00...... ------------------
18:00...... ------------------
19:00...... The Vogon reads his poem
20:00...... ------------------
20:30-22:15 Marvin escorts the Hitchhikers to the bridge
```

The time grid can be turned on and off with the variable 変数`org-agenda-use-time-grid` を使って and can be configured 設定することができます with `org-agenda-time-grid` を使って

10.4.3 Sorting of agenda items

Before being inserted into a view, the items are sorted. How this is done depends on the type of view.

- For the daily/weekly agenda, the items for each day are sorted. The default order デフォルトの順番は is to first collect all items containing 含んでいる an explicit time-of-day specification. These entries are shown at the beginning of the list, as a *schedule* for the day. After that, items remain grouped in categories, in the sequence given by `org-agenda-files`. Within each category, items are sorted by priority（節 5.4 [Priorities], ページ 58, を見てください）、which is composed of the base priority (2000 for priority '`A`', 1000 for '`B`', and 0 for '`C`'), plus additional increments for overdue scheduled or deadline items.
- For the TODO list, items remain in the order of categories, but within each category, sorting ソートは takes place 起きます according 従って to priority 優先順位に（節 5.4 [Priorities], ページ 58, を見てください）。The priority used for sorting derives from the priority cookie, with additions depending on 依存して how close どれくらい近いのか an item is to its due or scheduled date.
- For tags matches, items are not sorted at all, but just appear in the sequence in which they are found in the agenda files.

Sorting can be customized カスタマイズすることができます using 使って the variable 変数`org-agenda-sorting-strategy` を and may also include criteria based on 基いて the estimated effort of an entry（節 8.5 [Effort Estimates], ページ 89, を見てください）。

10.4.4 Filtering/limiting agenda times

Agenda built-in or customized commands are statically defined. Agenda filters and limits provide two ways of dynamically narrowing down the list of agenda entries: *filters* and *limits*. Filters only act on the display of the items, while limits take effect before the list of agenda entries is built. Filters are more often used interactively, インタラクティブに while

limits are mostly 主に useful 役に立ちます when defined as local variables within custom agenda commands.

Filtering in the agenda

`/` (`org-agenda-filter-by-tag`)

Filter the agenda view with respect to a tag and/or effort estimates. The difference between this and a custom agenda command is that filtering is very fast, so that you can switch quickly 素早く between different filters without having to recreate 再作成する必用なく the agenda. アジェンダを[6]

You are prompted プロンプトが出ます for a tag selection letter; `SPC` means any tag at all. Pressing `TAB` を at that prompt offers 提供します completion 補完を to select a tag, タグを選択するための including 含む any tags that do not have a selection character. The command then hides all entries that do not contain or inherit this tag. When called 呼ばれるときには with prefix argument, remove the entries that *do* have the tag. A second `/` at the prompt turns off the filter and shows 表示しまづ any hidden entries. Pressing `+` or `-` switches between filtering and excluding the next tag.

Org also supports サポートしています automatic, context-aware tag filtering. If the variable 変数`org-agenda-auto-exclude-function` is set to a user-defined function, that function can decide which tags should be excluded from the agenda automatically. 自動的に Once this is set, the `/` command then accepts `RET` as a sub-option key and runs the auto exclusion logic. For example, 例えば let's say しましょう you use a '`Net`' tag to identify tasks which need network access, an '`Errand`' tag for errands in town, and a '`Call`' tag for making phone calls. You could auto-exclude these tags based on the availability of the Internet, and outside of business hours, with something like this: このようなものを使って

```
(defun org-my-auto-exclude-function (tag)
  (and (cond
        ((string= tag "Net")
         (/= 0 (call-process "/sbin/ping" nil nil nil
                             "-c1" "-q" "-t1" "mail.gnu.org")))
        ((or (string= tag "Errand") (string= tag "Call"))
         (let ((hour (nth 2 (decode-time))))
           (or (< hour 8) (> hour 21)))))
       (concat "-" tag)))

(setq org-agenda-auto-exclude-function 'org-my-auto-exclude-function)
```

`<` (`org-agenda-filter-by-category`)

Filter the current agenda view with respect to the category of the item at point. Pressing `<` another time removes 取り除きます this filter. このフィルターを

[6] Custom commands can preset a filter by binding the variable 変数`org-agenda-tag-filter-preset` を as an option. オプションとして This filter is then applied to the view and persists as a basic filter through refreshes and more secondary filtering. The filter is a global property of the entire agenda view---in a block agenda, you should only set 設定するべきです this これを in the global options section, not in the section of an individual block.

When called with a prefix argument 1 つの前置引数といっしょに呼ばれるときには、exclude 除外します the category カテゴリーを of the item at point from the agenda.

You can add 追加することができます a filter preset フィルターのプリセットを in custom agenda commands through the option `org-agenda-category-filter-preset`. 節 10.6.3 [Setting options], ページ 132, を見てください。

^ (`org-agenda-filter-by-top-headline`)
: Filter the current agenda view and only display the siblings and the parent headline of the one at point.

= (`org-agenda-filter-by-regexp`)
: Filter the agenda view by a regular expression: only show agenda entries matching the regular expression the user entered. When called with a prefix argument, 1 つの前置引数といっしょに呼ばれうときには it filters *out* entries matching the regexp. Called in a regexp-filtered agenda view, remove the filter, unless there are two universal prefix arguments, in which case filters are cumulated.

  You can add 追加することができます a filter preset フィルターのプリセットを in custom agenda commands through the option `org-agenda-regexp-filter-preset`. 節 10.6.3 [Setting options], ページ 132, を見てください。

_ (`org-agenda-filter-by-effort`)
: Filter the agenda view with respect to effort estimates. You first need to set up allowed efforts globally, for example 例えば

```
(setq org-global-properties
      '(("Effort_ALL". "0 0:10 0:30 1:00 2:00 3:00 4:00")))
```

  You can then filter for an effort by first typing an operator, one of `<`, `>` and `=`, and then the one-digit index of an effort estimate in your array of allowed values, where *`0`* means the 10th value. The filter then restricts to entries with effort smaller-or-equal, equal, or larger-or-equal than the selected value. For application of the operator, entries without a defined effort 定義されている工数見積もりのない are treated according 従って to the value 値に of `org-sort-agenda-noeffort-is-high` の

  When called with a prefix argument, 1 つの前置引数といっしょに呼ばれるときには it それは removes 取り除きます entries エントリーを matching the condition. この条件にマッチする With two universal prefix arguments, it それは clears effort filters, which これ can be accumulated. 積み重なることがあります

  You can add 追加することができます a filter preset フィルターのプリセットを in custom agenda commands through the option `org-agenda-effort-filter-preset`. 節 10.6.3 [Setting options], ページ 132, を見てください。

| (`org-agenda-filter-remove-all`)
: Remove all filters in the current agenda view.

Setting limits for the agenda

Here これは is a list of options オプションのリストです that you can set, either globally, or locally in your custom agenda views（節 10.6 [Custom Agenda Views], ページ 130, を見てください）

`org-agenda-max-entries`
: Limit the number of entries.

`org-agenda-max-effort`
: Limit the duration of accumulated efforts (as minutes).

`org-agenda-max-todos`
: Limit the number of entries with TODO keywords.

`org-agenda-max-tags`
: Limit the number of tagged entries.

When set 設定されているときには、to a positive integer, 正の整数に each option excludes 除外します entries from other categories: for example, 例えば'`(setq org-agenda-max-effort 100)`' は limits the agenda to 100 minutes of effort and exclude any entry that has no effort property. If you want to include entries with no effort property, use a negative value for `org-agenda-max-effort`. One useful setup is to use `org-agenda-max-entries` を locally in a custom command. For example, 例えば this custom command displays 表示します the next five entries with a '`NEXT`' TODO keyword.

```
(setq org-agenda-custom-commands
      '(("n" todo "NEXT"
         ((org-agenda-max-entries 5)))))
```

Once you mark one of these five entry as DONE, DONE として rebuilding the agenda will again the next five entries again, including 含む the first entry that was excluded so far.

You can also dynamically set temporary limits, which これらは are lost 失なわれます when rebuilding the agenda: アジェンダを再ビルドするちんいに

`~` (`org-agenda-limit-interactively`)
: This これは prompts プロンプトを出します for the type of limit to apply and its value.

10.5 Commands in the Agenda Buffer

Entries in the agenda buffer are linked back to the Org file or diary file where they originate. You are not allowed to edit the agenda buffer itself, but commands are provided to show and jump to the original entry location, and to edit the Org files "remotely" from the agenda buffer. In this way, all information is stored only once, removing the risk that your agenda and note files may diverge.

Some commands can be executed with mouse clicks on agenda lines. For the other commands, point ポイントが needs to be in the desired line.

Motion

n (`org-agenda-next-line`)
: Next line (same as `DOWN` and *C-n*).

p (org-agenda-previous-line)
: Previous line (same as UP and *C-p*).

View/Go to Org file

SPC or *mouse-3* (org-agenda-show-and-scroll-up)
: Display the original location of the item in another window. With a prefix argument, 1 つの前置引数といっしょだと make sure 確実にしあますう that drawers stay folded.

L (org-agenda-recenter)
: Display original location and recenter that window.

TAB or *mouse-2* (org-agenda-goto)
: Go to the original location of the item in another window.

RET (org-agenda-switch-to)
: Go to the original location of the item and delete other windows.

F (org-agenda-follow-mode)
: Toggle トグルします Follow モードを In Follow mode, as you move point through the agenda buffer, the other window always 常に shows 表示します the corresponding location 対応する場所を in the Org file. The initial setting for this mode in new agenda buffers can be set 設定することができます with the 変数org-agenda-start-with-follow-mode を使って

C-c C-x b (org-agenda-tree-to-indirect-buffer)
: Display the entire subtree of the current item in an indirect buffer. With a numeric prefix argument N, go up to level N and then take that tree. If N is negative, go up that many levels. With a *C-u* prefix, do not remove the previously used indirect buffer.

C-c C-o (org-agenda-open-link)
: Follow a link in the entry. This これは offers 提供します a selection of any links in the text belonging to the referenced Org node. If there is only one link, follow it それをたどります without a selection prompt. 選択プロンプトなしで

Change display

A
: Interactively インタラクティブに select another agenda view and append it to the current view.

o
: Delete other windows.

v d or short *d* (org-agenda-day-view)
: Switch to day view. When switching to day view, this setting becomes the default for subsequent agenda refreshes. A numeric prefix argument may be used to jump directly to a specific day of the year. For example, 例えば*32 d* は jumps to February 1st. When setting day view, a year may be encoded in the prefix argument as well. For example, 例えば*200712 d* は jumps to January 12, 2007. If such a year specification has only one or two digits, it is expanded into one of the 30 next years or the last 69 years.

`v w` or short `w` (`org-agenda-week-view`)
: Switch to week view. When switching week view, this setting becomes the default for subsequent agenda refreshes. A numeric prefix argument may be used to jump directly to a specific day of the ISO week. For example 例えば`9 w` は to ISO week number 9. When setting week view, a year may be encoded in the prefix argument as well. For example, 例えば`200712 w` は jumps to week 12 in 2007. If such a year specification has only one or two digits, it is expanded into one of the 30 next years or the last 69 years.

`v m` (`org-agenda-month-view`)
: Switch to month view. Because month views are slow to create, they do not become the default for subsequent agenda refreshes. A numeric prefix argument may be used to jump directly to a specific day of the month. When setting month view, a year may be encoded in the prefix argument as well. For example, 例えば`200712 m` は jumps to December, 2007. If such a year specification has only one or two digits, it is expanded into one of the 30 next years or the last 69 years.

`v y` (`org-agenda-year-view`)
: Switch to year view. Because year views are slow to create, they do not become the default for subsequent agenda refreshes. A numeric prefix argument may be used to jump directly to a specific day of the year.

`v SPC` (`org-agenda-reset-view`)
: Reset the current view to `org-agenda-span`.

`f` (`org-agenda-later`)
: Go forward in time to display the span following the current one. For example, 例えば if the display covers a week, switch to the following week. With a prefix argument, 1 つの前置引数といっしょだと repeat that many times.

`b` (`org-agenda-earlier`)
: Go backward in time to display earlier dates.

`.` (`org-agenda-goto-today`)
: Go to today.

`j` (`org-agenda-goto-date`)
: Prompt for a date and go there.

`J` (`org-agenda-clock-goto`)
: Go to the currently clocked-in task *in the agenda buffer*.

`D` (`org-agenda-toggle-diary`)
: Toggle トグルします the inclusion 含めるかどうかを of diary entries. 節 10.3.1 [Weekly/daily agenda], ページ 110, を見てください。

`v l` or `v L` or short `l` (`org-agenda-log-mode`)
: Toggle Logbook mode. In Logbook mode, entries that were marked DONE while logging was on（変数`org-log-done` を見てください）are shown in the agenda, as are entries that have been clocked on that day. You can configure 設定することができます the entry types that should be included in log mode

using the variable 変数`org-agenda-log-mode-items` を使って When called 呼ばれるときには with a `C-u` prefix, show all possible logbook entries, including 含む state changes. When called 呼ばれるときには with two prefix arguments `C-u C-u`, show only logging information, nothing else. `v L` is equivalent to `C-u v l`.

`v [` or short `[` (`org-agenda-manipulate-query-add`)
: Include inactive timestamps into the current view. Only for weekly/daily agenda.

`v a` (`org-agenda-archives-mode`)
: Toggle Archives mode. In Archives mode, trees that are archived（節 9.6.2 [Internal archiving], ページ 106, を見てください）are also scanned when producing the agenda. To exit 抜けるには archives mode, press `v a` again.

`v A`
: Toggle トグルします Archives mode. Include all archive files as well.

`v R` or short `R` (`org-agenda-clockreport-mode`)
: Toggle トグルします Clockreport mode. In Clockreport mode, the daily/weekly agenda always 常に shows 表示します a table with the clocked times for the time span and file scope covered by the current agenda view. The initial setting for this mode in new agenda buffers can be set with the variable 変数`org-agenda-start-with-clockreport-mode`. By using 使うことによって a prefix argument 前置引数を when toggling this mode (i.e., すなわち`C-u R`), the clock table does not show contributions from entries that are hidden by agenda filtering[7]. 変数`org-clock-report-include-clocking-task` も見てください。

`v c`
: Show overlapping clock entries, clocking gaps, and other clocking problems in the current agenda range. You can then visit clocking lines and fix them manually. See 見てください the variable 変数`org-agenda-clock-consistency-checks` for information on how to customize カスタマイズする方法についての the definition 定義を of what constituted a clocking problem. To return 戻るには to normal agenda display, press `l` to exit Logbook mode.

`v E` or short `E` (`org-agenda-entry-text-mode`)
: Toggle トグルします entry text mode. In entry text mode, a number of lines from the Org outline node referenced by an agenda line are displayed below the line. The maximum number of lines is given by the variable 変数`org-agenda-entry-text-maxlines`. Calling this command with a numeric prefix argument temporarily modifies that number to the prefix value.

`G` (`org-agenda-toggle-time-grid`)
: Toggle トグルします the time grid 時間のグリッドの on and off. オンオフを See 見てください。also the variables 変数`org-agenda-use-time-grid` と`org-agenda-time-grid` を

`r` (`org-agenda-redo`)
`g`
: Recreate the agenda buffer, for example 例えば to reflect the changes after modification of the timestamps of items with `S-LEFT` and `S-RIGHT`. When the

[7] Only tags filtering is respected here, effort filtering is ignored.

buffer is the global TODO list, a prefix argument is interpreted to create a selective list for a specific TODO keyword.

C-x C-s or short *s* (`org-save-all-org-buffers`)
Save all Org buffers in the current Emacs session, and also the locations of IDs.

C-c C-x C-c (`org-agenda-columns`)
Invoke column view（節 7.5 [Column View], ページ 70, を見てください）in the agenda buffer. The column view format is taken from the entry at point, or, if there is no entry at point, from the first entry in the agenda view. So whatever the format for that entry would be in the original buffer (taken from a property, from a 'COLUMNS' keyword, or from the default variable 変数`org-columns-default-format`) is used in the agenda.

C-c C-x > (`org-agenda-remove-restriction-lock`)
Remove the restriction lock on the agenda, if it is currently restricted to a file or subtree（節 10.1 [Agenda Files], ページ 108, を見てください）。

M-UP (`org-agenda-drag-line-backward`)
Drag the line at point backward one line. With a numeric prefix argument, drag backward by that many lines.

Moving agenda lines does not persist after an agenda refresh and does not modify the contributing Org files.

M-DOWN (`org-agenda-drag-line-forward`)
Drag the line at point forward one line. With a numeric prefix argument, drag forward by that many lines.

Remote editing

0--9 Digit argument.

C-_ (`org-agenda-undo`)
Undo a change due to a remote editing command. The change is undone both in the agenda buffer and in the remote buffer.

t (`org-agenda-todo`)
Change the TODO state of the item, both in the agenda and in the original Org file.

C-S-RIGHT (`org-agenda-todo-nextset`)
Switch to the next set of TODO keywords.

C-S-LEFT, `org-agenda-todo-previousset`
Switch to the previous set of TODO keywords.

C-k (`org-agenda-kill`)
Delete 削除します the current agenda item along with ともに the entire subtree belonging to it in the original Org file. If the text to be deleted remotely is longer than one line, the kill このキルは needs to be confirmed 確認される必用があります by the user. ユーザーによって変数`org-agenda-confirm-kill`を見てください。

C-c C-w (`org-agenda-refile`)
: Refile the entry at point.

C-c C-x C-a or short `a` (`org-agenda-archive-default-with-confirmation`)
: Archive アーカイブします the subtree サブツリーを corresponding 対応する to the entry エントリーに at point using 使って the default archiving command デフォルトのアーカイブコマンドを set 設定されている in `org-archive-default-command` の中で When using 使うときには the `a` キーを confirmation is required. 確認が必用です

C-c C-x a (`org-agenda-toggle-archive-tag`)
: Toggle トグルします the archive tag（節 9.6.2 [Internal archiving], ページ 106, を見てください）for the current headline.

C-c C-x A (`org-agenda-archive-to-archive-sibling`)
: Move 移動します the subtree corresponding to the current entry to its *archive sibling*.

C-c C-x C-s or short *$* (`org-agenda-archive`)
: Archive アーカイブします the subtree サブツリーを corresponding 対応する to the current headline. 現在の見出しに This これは means 意味します the entry このエントリーが is moved to the configured archive location, most likely ほとんどの場合に a different file. 異なるファイルへ

T (`org-agenda-show-tags`)
: Show all tags associated with the current item. This is useful これが役に立ちます if you have turned off `org-agenda-show-inherited-tags`, but still want to see all tags of a headline occasionally.

: (`org-agenda-set-tags`)
: Set tags for the current headline. If there is an active region in the agenda, change a tag for all headings in the region.

, (`org-agenda-priority`)
: Set the priority for the current item. Org-mode prompts プロンプトを出します for the priority character. If you reply with `SPC`, the priority cookie is removed from the entry.

P (`org-agenda-show-priority`)
: Display weighted priority of current item.

+ or *S-UP* (`org-agenda-priority-up`)
: Increase the priority of the current item. The priority is changed in the original buffer, but the agenda is not resorted. Use 使います the *r* key for this.

- or *S-DOWN* (`org-agenda-priority-down`)
: Decrease the priority of the current item.

C-c C-z or short *z* (`org-agenda-add-note`)
: Add a note to the entry. This note is recorded, and then filed to the same location where state change notes are put. Depending on `org-log-into-drawer` に依存して this これは may be inside a drawer.

C-c C-a (org-attach)
: Dispatcher for all command related to attachments.

C-c C-s (org-agenda-schedule)
: Schedule this item. With a prefix argument, 1 つの前置引数といっしょだと remove 取り除きあますう the scheduling timestamp

C-c C-d (org-agenda-deadline)
: Set a deadline for this item. With a prefix argument, 1 つの前置引数といっしょだしたお remove 取り除きます。the deadline.

S-RIGHT (org-agenda-do-date-later)
: Change the timestamp associated with the current line by one day into the future. If the date is in the past, the first call to this command moves it to today. With a numeric prefix argument, change it by that many days. For example, 例えば*3 6 5 S-RIGHT* は changes it by a year. With a *C-u* 前置引数といっしょだと change the time by one hour. If you immediately repeat the command, このコマンドを it will continue to change hours even without the prefix argument. 前置引数がないとしても With a double *C-u C-u* prefix, do the same for changing minutes. The stamp is changed in the original Org file, but the change is not directly reflected in the agenda buffer. Use 使ってくださいい*r* or *g* to update the buffer.

S-LEFT (org-agenda-do-date-earlier)
: Change the timestamp associated with the current line by one day into the past.

> (org-agenda-date-prompt)
: Change the timestamp associated with the current line. The key > has been chosen, because it is the same as *S-.* on my keyboard.

I (org-agenda-clock-in)
: Start the clock on the current item. If a clock is running already, it is stopped first.

O (org-agenda-clock-out)
: Stop the previously started clock.

X (org-agenda-clock-cancel)
: Cancel the currently running clock.

J (org-agenda-clock-goto)
: Jump ジャンプします to the running clock in another window.

k (org-agenda-capture)
: Like org-capture に似ていますが but use 使います the date 日付を at point ポイントにある as the default date for the capture template. これをorg-capture のデフォルトのふるまいにするには、org-capture-use-agenda-date を見てください。

Bulk remote editing selected entries

m (`org-agenda-bulk-mark`)
: Mark the entry at point for bulk action. If there is an active region in the agenda, mark the entries in the region. With numeric prefix argument, mark that many successive entries.

*** (`org-agenda-bulk-mark-all`)
: Mark all visible agenda entries for bulk action.

u (`org-agenda-bulk-unmark`)
: Unmark entry for bulk action.

U (`org-agenda-bulk-remove-all-marks`)
: Unmark all marked entries for bulk action.

M-m (`org-agenda-bulk-toggle`)
: Toggle トグルします mark of the entry at point for bulk action.

*M-** (`org-agenda-bulk-toggle-all`)
: Toggle トグルします mark of every entry for bulk action.

% (`org-agenda-bulk-mark-regexp`)
: Mark entries matching a regular expression for bulk action.

B (`org-agenda-bulk-action`)
: Bulk action: act on all marked entries in the agenda. This prompts プロンプトを出します for another key to select the action to be applied. The prefix argument to *B* is passed through to the *s* and *d* commands, to bulk-remove these special timestamps. By default, デフォルトで marks マークは are removed 取り除かれます after the bulk. If you want them to persist, set `org-agenda-bulk-persistent-marks` to `t` or hit *p* at the prompt.

  *** Toggle トグルします persistent marks.

  $ Archive アーカイブします

  all selected entries. 選択されている全てのエントリーを

  A Archive アーカイブします entries エントリーを by moving 移動することによって them それを to their respective archive siblings.

  t Change TODO state. This prompts プロンプトを出します for a single TODO keyword and changes the state of all selected entries, bypassing blocking and suppressing logging notes---but not timestamps.

  + Add a tag to all selected entries.

  - Remove a tag from all selected entries.

  s Schedule all items to a new date. To shift existing schedule dates by a fixed number of days, use something starting with double plus at the prompt, for example 例えば‘`++8d`’ or ‘`++2w`’.

  d Set deadline to a specific date.

`r` Prompt for a single refile target and move all entries. The entries are no longer in the agenda; refresh (`g`) to bring them back.

`S` Reschedule randomly into the coming N days. N is prompted for. With a prefix argument 1 つの前置引数といっしょだと(`C-u B S`)、scatter only across weekdays.

`f` Apply a function[8] to marked entries. For example, 例えば the function below sets the 'CATEGORY' property of the entries to 'web'.

```
(defun set-category ()
  (interactive "P")
  (let ((marker (or (org-get-at-bol 'org-hd-marker)
                    (org-agenda-error))))
    (org-with-point-at marker
      (org-back-to-heading t)
      (org-set-property "CATEGORY" "web"))))
```

Calendar commands

`c` (`org-agenda-goto-calendar`)
Open the Emacs calendar and go to the date at point in the agenda.

`c` (`org-calendar-goto-agenda`)
When in the calendar, compute and show the Org agenda for the date at point.

`i` (`org-agenda-diary-entry`)
Insert 挿入します a new entry into the diary, using the date at point and (for block entries) the date at the mark. This adds to the Emacs diary file[9], in a way similar to the `i` command in the calendar. The diary file pops up in another window, where you can add 追加することができます the entry.

If you configure 設定したら`org-agenda-diary-file` を to point 指すように to an Org file, Org-mode のファイルを Org Org-mode は creates 作成します entries エントリーを in that file instead. かわりに Most entries are stored in a date-based outline tree that will later make it easy to archive appointments from previous months/years. The tree is built under an entry with a 'DATE_TREE' property, or else with years as top-level entries. Emacs prompts プロンプトを出します you for the entry text ---if you specify it, the entry is created 作成されます in `org-agenda-diary-file` without further interaction. If you directly press `RET` at the prompt without typing text, the target file is shown in another window for you to finish the entry there. `k r` コマンドも見てください。

`M` (`org-agenda-phases-of-moon`)
Show the phases of the moon for the three months around current date.

`S` (`org-agenda-sunrise-sunset`)
Show 表示します sunrise and sunset times. The geographical location 地理的な場所を must be set 設定されなければなりません with calendar variables, see the documentation for the Emacs calendar.

[8] You can also create 作成することもできます persistent custom functions through `org-agenda-bulk-custom-functions` をとおして

[9] This file is parsed for the agenda when `org-agenda-include-diary` is set.

`C` (`org-agenda-convert-date`)
: Convert the date at point into many other cultural and historic calendars.

`H` (`org-agenda-holidays`)
: Show holidays for three months around point date.

Quit and exit

`q` (`org-agenda-quit`)
: Quit agenda, remove the agenda buffer.

`x` (`org-agenda-exit`)
: Exit agenda, remove the agenda buffer and all buffers loaded by Emacs for the compilation of the agenda. Buffers created by the user to visit Org files are not removed.

10.6 Custom Agenda Views

Custom agenda commands serve two purposes: to store and quickly 素早く access frequently used TODO and tags searches, and to create special composite agenda buffers. Custom agenda commands are accessible through the dispatcher（節 10.2 [Agenda Dispatcher], ページ 108, を見てください）、just like the default commands.

10.6.1 Storing searches

The first application of custom searches is the definition of keyboard shortcuts for frequently used searches, either creating an agenda buffer, or a sparse tree (the latter covering of course only the current buffer).

Custom commands カスタムコマンドは are configured 設定されます。in the variable 変数`org-agenda-custom-commands` の中で You can customize カスタマイズすることができます this variable, この変数を for example 例えば by pressing `C` を押すことににによって from the agenda dispatcher（節 10.2 [Agenda Dispatcher], ページ 108, を見てください）。You can also directly set it with Emacs Lisp in the Emacs init file. The following example contains all valid agenda views:

```
(setq org-agenda-custom-commands
      '(("x" agenda)
        ("y" agenda*)
        ("w" todo "WAITING")
        ("W" todo-tree "WAITING")
        ("u" tags "+boss-urgent")
        ("v" tags-todo "+boss-urgent")
        ("U" tags-tree "+boss-urgent")
        ("f" occur-tree "\\<FIXME\\>")
        ("h" . "HOME+Name tags searches") ;description for "h" prefix
        ("hl" tags "+home+Lisa")
        ("hp" tags "+home+Peter")
        ("hk" tags "+home+Kim")))
```

The initial string in each entry defines 定義します the keys you have to press after the dispatcher command in order to access the command. Usually 通常は this これは will

be just a single character, but if you have many similar commands, you can also define two-letter combinations where the first character is the same in several combinations and serves as a prefix key[10]. The second parameter is the search type, followed by the string or regular expression to be used for the matching. The example above will therefore define:

`x` as a global search for agenda entries planned[11] this week/day.

`y` as the same search, but only for entries with an hour specification like '`[h]h:mm`'---think of them as appointments.

`w` as a global search for TODO entries with '`WAITING`' as the TODO keyword.

`W` as the same search, but only in the current buffer and displaying the results as a sparse tree.

`u` as a global tags search for headlines tagged '`boss`' but not '`urgent`'.

`v` The same search, but limiting it to headlines that are also TODO items.

`U` as the same search, but only in the current buffer and displaying the result as a sparse tree.

`f` to create a sparse tree スパースツリーを(again, current buffer only) with all entries containing 含んでいる the word '`FIXME`' という単語を

`h` as a prefix command for a '`HOME`' tags search where you have to press an additional key (`l`, `p` or `k`) to select a name (Lisa, Peter, or Kim) as additional tag to match.

Note 注意してください that `*-tree` agenda views need to be called 呼ばれる必用があります from an Org buffer as they operate on the current buffer only.

10.6.2 Block agenda

Another possibility is the construction of agenda views that comprise the results of *several* commands, each of which creates 作成します a block in the agenda buffer. The available commands include `agenda` for the daily or weekly agenda (as created with `a`) , `alltodo` for the global TODO list (as constructed with `t`), and the matching commands discussed above: `todo`, `tags`, and `tags-todo`. Here are two examples: これらは 2 つの例です:

```
(setq org-agenda-custom-commands
      '(("h" "Agenda and Home-related tasks"
         ((agenda "")
          (tags-todo "home")
          (tags "garden")))
        ("o" "Agenda and Office-related tasks"
         ((agenda "")
          (tags-todo "work")
          (tags "office")))))
```

[10] You can provide a description for a prefix key by inserting a cons cell with the prefix and the description.

[11] *Planned* means here that these entries have some planning information attached to them, like a time-stamp, a scheduled or a deadline string. See `org-agenda-entry-types` on how to set what planning information is taken into account.

This これは defines 定義しますh を to create 作成するための a multi-block view for stuff you need to attend 出席する必用がある to at home. 家で The resulting agenda buffer contains your agenda for the current week, all TODO items that carry the tag 'home', and also all lines tagged with 'garden'. Finally 最後に the command o provides 提供しています a similar view 似たようなビューを for office tasks.

10.6.3 Setting options for custom commands

Org-mode contains a number of variables regulating agenda construction and display. The global variables define the behavior for all agenda commands, including 含む the custom commands. However, しかし if you want to change 変更したいのであれば some settings just for a single custom view, you can do so. Setting options requires inserting a list of variable names and values at the right spot in org-agenda-custom-commands. 例です:

```
(setq org-agenda-custom-commands
      '(("w" todo "WAITING"
         ((org-agenda-sorting-strategy '(priority-down))
          (org-agenda-prefix-format "  Mixed: ")))
        ("U" tags-tree "+boss-urgent"
         ((org-show-context-detail 'minimal)))
        ("N" search ""
         ((org-agenda-files '("~org/notes.org"))
          (org-agenda-text-search-extra-files nil)))))
```

Now the w command sorts the collected entries only by priority, and the prefix format is modified to just say 'Mixed:' instead かわりに of giving 与える the category カテゴリーを of the entry. エントリーの The sparse tags tree of U now turns out ultra-compact, because neither the headline hierarchy above the match, nor the headline following the match are shown. The command N does a text search limited to only a single file.

For command sets creating a block agenda, org-agenda-custom-commands has two separate spots for setting options. You can add 追加することができます options オプションを that should be valid for just a single command in the set, and options that should be valid for all commands in the set. The former are just added to the command entry; the latter must come after the list of command entries. Going back to the block agenda example（節 10.6.2 [Block agenda], ページ 131, を見てください）、let's change the sorting strategy for the h commands to priority-down, but let's sort the results for 'garden' tags query in the opposite order, priority-up. これはこのように見えます:

```
(setq org-agenda-custom-commands
      '(("h" "Agenda and Home-related tasks"
         ((agenda)
          (tags-todo "home")
          (tags "garden"
                ((org-agenda-sorting-strategy '(priority-up)))))
         ((org-agenda-sorting-strategy '(priority-down))))
        ("o" "Agenda and Office-related tasks"
         ((agenda)
          (tags-todo "work")
          (tags "office")))))
```

As you see, the values and parentheses setting is a little complex. When in doubt, use 使ってください the customize interface to set 設定するには this variable この変数を---it それは fully supports サポートしています its structure. その構造を Just one caveat: when setting options in this interface, the *values* are just Lisp expressions. So if the value is a string, you need to add 追加する必用があります the double-quotes around the value yourself. 自分自身で

To control コントロールするには whether an agenda command should be accessible from a specific context, you can customize カスタマイズすることができます`org-agenda-custom-commands-contexts` を Let's say しましょう for example 例えば that you have an agenda command `o` displaying a view that you only need when reading emails. Then そうすると、you would configure 設定します this option このオプションを like this: このようにして

```
(setq org-agenda-custom-commands-contexts
      '(("o" (in-mode . "message-mode"))))
```

You can also tell that the command key `o` should refer to another command key `r`. In that case, その場合には add 追加してください: this command key このコマンドキーを like this: このようにして

```
(setq org-agenda-custom-commands-contexts
      '(("o" "r" (in-mode . "message-mode"))))
```

さらに情報については、この変数のドキュメント文字列を見てください。

10.7 Exporting Agenda Views

If you are away from your computer, it can be very useful とても役に立つことがあります to have a printed version of some agenda views to carry around. Org-mode can export custom agenda views as plain text, HTML[12], Postscript, PDF[13], and iCalendar files. If you want to do this only occasionally, use the following command:

`C-x C-w` (`org-agenda-write`)
: Write the agenda view to a file.

If you need to export エクスポートする必用があれば certain agenda views frequently, 頻繁に you can associate 割り当てることができます any custom agenda command with a list of output file names[14]. Here これは is an example 例です that first 最初に defines 定義します custom commands for the agenda and the global TODO list, together with a number of files to which to export them. Then we define two block agenda commands and specify file names for them as well. File names can be relative to the current working directory, or absolute.

```
(setq org-agenda-custom-commands
      '(("X" agenda "" nil ("agenda.html" "agenda.ps"))
        ("Y" alltodo "" nil ("todo.html" "todo.txt" "todo.ps"))
```

[12] For HTML you need to install インストールする必用があ Hrvoje います Niksic の'`htmlize.el`' を from Hrvoje Niksic's repository (`https://github.com/hniksic/emacs-htmlize`) から

[13] To create 作成するには PDF の出力を the Ghostscript ps2pdf utility must be installed インストールされていなければなりません on the system. Selecting a PDF file also creates 作成します the postscript file.

[14] If you want to store standard views like the weekly agenda or the global TODO list as well, you need to define 定義する必用があります custom commands for them in order to be able to specify file names.

```
("h" "Agenda and Home-related tasks"
 ((agenda "")
  (tags-todo "home")
  (tags "garden"))
 nil
 ("~/views/home.html"))
("o" "Agenda and Office-related tasks"
 ((agenda)
  (tags-todo "work")
  (tags "office"))
 nil
 ("~/views/office.ps" "~/calendars/office.ics"))))
```

The extension of the file name determines the type of export. If it is '`.html`', Org-mode は uses 使います the htmlize package to convert the buffer to HTML and save it to this file name. If the extension is '`.ps`', `ps-print-buffer-with-faces` is used to produce Postscript output. If the extension is '`.ics`', iCalendar export is run export over all files that were used to construct the agenda, and limit the export to entries listed in the agenda. Any other extension produces a plain ASCII file.

The export files are *not* created when you use one of those commands interactively インタラクティブに because this might use too much overhead. Instead, かわりに there is a special command 特別なコマンドがあります to produce 生成するための*all* specified files in one step:

`e` (`org-store-agenda-views`)
: Export エクスポートします all agenda views 全てのアジェンダビューを that have 持っている export file names associated with them.

You can use 使うことができます the options section オプションセクションを of the custom agenda commands カスタムアジェンダコマンドの to also set 設定するためにも options オプションを for the export commands. 例です:

```
(setq org-agenda-custom-commands
      '(("X" agenda ""
         ((ps-number-of-columns 2)
          (ps-landscape-mode t)
          (org-agenda-prefix-format " [ ] ")
          (org-agenda-with-colors nil)
          (org-agenda-remove-tags t))
         ("theagenda.ps"))))
```

This command このコマンドは sets two options for the Postscript exporter, to make it print in two columns in landscape format---the resulting page can be cut in two and then used in a paper agenda. The remaining settings modify the agenda prefix to omit 省略するように category and scheduling information, and instead かわりに include 含める a checkbox to check off items. We also remove the tags to make the lines compact, and we do not want to use colors for the black-and-white printer. Settings specified in `org-agenda-exporter-settings` also apply, e.g., 例えば

```
(setq org-agenda-exporter-settings
      '((ps-number-of-columns 2)
```

```
          (ps-landscape-mode t)
          (org-agenda-add-entry-text-maxlines 5)
          (htmlize-output-type 'css)))
```

but the settings in `org-agenda-custom-commands` take precedence.

From the command line you may also use:

```
emacs -eval (org-batch-store-agenda-views) -kill
```

or, if you need to modify 変更する必用があれば some parameters[15]

```
emacs -eval '(org-batch-store-agenda-views                      \
              org-agenda-span (quote month)                     \
              org-agenda-start-day "2007-11-01"                 \
              org-agenda-include-diary nil                      \
              org-agenda-files (quote ("~/org/project.org")))'  \
      -kill
```

which これは creates 作成します the agenda views restricted to the file '`~/org/project.org`', without diary entries and with a 30-day extent.

You can also extract agenda information in a way that allows further processing by other programs. See 節 A.9 [Extracting Agenda Information], ページ 266, を見てください。for more information. さらに情報については、

10.8 Using Column View in the Agenda

Column view 列ビューは（節 7.5 [Column View], ページ 70, を見てください）is normally 通常は used 使われあますう to view and edit properties embedded in the hierarchical structure of an Org file. It can be quite useful 極めて役に立つことがあります to use column view also from the agenda, where entries are collected by certain criteria.

C-c C-x C-c (`org-agenda-columns`)
: Turn on column view in the agenda.

To understand 理解するには how to use this properly, it is important 重要です to realize that the entries in the agenda are no longer in their proper outline environment. これは次の問題を起こします:

1. Org needs to make a decision 決める必用があります which columns format to use. Since the entries in the agenda are collected from different files, and different files may have different columns formats, this is a non-trivial problem. Org first checks if the variable 変数`org-agenda-overriding-columns-format` is currently set, and if so, takes the format from there. そこから Otherwise そうでなければ it takes the format associated with the first item in the agenda, or, if that item does not have a specific format (defined in a property, or in its file), it それは uses 使います`org-columns-default-format` を
2. If any of the columns has a summary type defined（節 7.5.1.2 [Column attributes], ページ 70, を見てください）turning on column view in the agenda visits all relevant agenda files and make sure that the computations of this property are up to date. This これは is also true for the special '`CLOCKSUM`' property. Org then sums the values displayed

[15] Quoting depends on the system you use, please どうか check チェックしてくだせいあい the FAQ for examples. 例については

in the agenda. In the daily/weekly agenda, the sums cover a single day; in all other views they cover the entire block.

It is important to realize that the agenda may show the same entry *twice*---for example 例えば as scheduled and as a deadline---and it may show two entries from the same hierarchy (for example 例えば a *parent* and its *child*). In these cases, the summation in the agenda leads to incorrect results because some values count double.

3. When the column view in the agenda shows 表示するときには the '`CLOCKSUM`' プロパティを that is always 常に the entire clocked time for this item. So even in the daily/weekly agenda, the clocksum listed in column view may originate from times outside the current view. This has the advantage that you can compare these values with a column listing the planned total effort for a task---one of the major applications for column view in the agenda. If you want information about clocked time in the displayed period use clock table mode (press `R` in the agenda).
4. When the column view in the agenda shows 表示するときにあ the '`CLOCKSUM_T`' property, that is always 常に today's clocked time for this item. So even in the weekly agenda, the clocksum listed in column view only originates from today. This これは lets 可能にします you compare the time you spent on a task for today, with the time already spent---via '`CLOCKSUM`'---and with the planned total effort for it.

11 Markup for Rich Contents

Org is primarily about organizing and searching through your plain-text notes. However, しかし it also provides 提供しています a lightweight yet robust markup language for rich text formatting and more. For instance, 例えば you may want to center or emphasize text. Or または you may need to insert 挿入する必用があるかもしれません a formula 数式や or image イメージを in your writing. 執筆することの中に Org Org-mode は offers 提供しています syntax 文法を for all of this and more. Used in conjunction with the export framework（章 12 [Exporting], ページ 145, を見てください）you can author beautiful documents in Org---like the fine manual you are currently reading.

11.1 Paragraphs

Paragraphs are separated by at least one empty line. If you need to enforce 強制する必用があれば a line break 改行を within a paragraph, use '`\\`' を使ってください at the end of a line. 行の終わりで

To preserve 保存し the line breaks, indentation and blank lines in a region, リージョンの中の but otherwise その他は use 使うのであれば normal formatting, 通常のフォーマットを you can use this construct, この文法を使うことができます which これを使うことができます。can also be used to format poetry. 詩をフォーマットするためにも

```
#+BEGIN_VERSE
 Great clouds overhead
 Tiny black birds rise and fall
 Snow covers Emacs

    ---AlexSchroeder
#+END_VERSE
```

When quoting a passage from another document, it is customary to format this as a paragraph that is indented on both the left and the right margin. You can include quotations in Org documents like this: このようにして

```
#+BEGIN_QUOTE
Everything should be made as simple as possible,
but not any simpler ---Albert Einstein
#+END_QUOTE
```

If you would like to center some text, do it それを行なってください like this: このようにして

```
#+BEGIN_CENTER
Everything should be made as simple as possible, \\
but not any simpler
#+END_CENTER
```

11.2 Emphasis and Monospace

You can make words '`*bold*`', '`/italic/`', '`_underlined_`', '`=verbatim=`' and '`~code~`', and, if you must, '`+strike-through+`'. Text in the code and verbatim string is not processed for Org-mode specific syntax; it is exported verbatim.

To turn off オフにするには fontification フォントづけを for marked up text, you can set `org-fontify-emphasized-text` to `nil`. To narrow down 狭めるには the list リストを of available markup syntax, you can customize `org-emphasis-alist` をカスタマイズすることができます。

11.3 Subscripts and Superscripts

'`^`' and '`_`' are used to indicate super- and subscripts. To increase 増やすため the readability of ASCII text, it is not necessary, but OK, to surround multi-character sub- and superscripts with curly braces. For example

```
The radius of the sun is R_sun = 6.96 x 10^8 m.  On the other hand,
the radius of Alpha Centauri is R_{Alpha Centauri} = 1.28 x R_{sun}.
```

If you write a text where the underscore is often used in a different context, Org's convention to always 常に interpret these as subscripts can get in your way. Configure the variable 変数`org-use-sub-superscripts` を設定してください。to change this convention. この規約を変更するには For example, 例えば when setting this variable to `{}`, '`a_b`' is not interpreted as a subscript, but '`a_{b}`' is.

*C-c C-x * (`org-toggle-pretty-entities`)
: This command formats sub- and superscripts in a WYSIWYM way.

11.4 Special Symbols

You can use 使うことができますLaTeX に似た文法を to insert 挿入するために special symbols ---named entities---like '`\alpha`' to indicate the Greek letter, or '`\to`' to indicate an arrow. Completion for these symbols is available, just type '`\`' and maybe a few letters, and press *M-TAB* to see possible completions. If you need 必用であれば such a symbol そのようなシンボルが inside a word, 単語の内側で terminate 終端して it それを with a pair of curly brackets. For example 例えば

```
Pro tip: Given a circle \Gamma of diameter d, the length of its
circumference is \pi{}d.
```

A large number of entities is provided, with names taken from both HTML and LaTeX; you can comfortably browse the complete list from a dedicated buffer using the command `org-entities-help`. It is also possible to provide your own special symbols in the variable `org-entities-user`.

During export, these symbols are transformed into the native format of the exporter back-end. Strings like '`\alpha`' are exported as '`&alpha;`' in the HTML output, and as '`\(\alpha\)`' in the LaTeX output. Similarly, '`\nbsp`' becomes '` `' in HTML and '`~`' in LaTeX.

If you would like to see entities displayed as UTF-8 characters, use the following command[1]:

*C-c C-x * (`org-toggle-pretty-entities`)
: Toggle トグルします display of entities as UTF-8 characters. This does not change the buffer content which remains plain ASCII, but it overlays the UTF-8 character for display purposes only.

[1] You can turn this on by default デフォルトで by setting 設定することによって the 変数`org-pretty-entities` を or on a per-file base ファイル毎には with the '`STARTUP`' option '`entitiespretty`'.

In addition 加えて to regular entities defined above, Org exports in a special way[2] the following commonly used character combinations: '`\-`' is treated as a shy hyphen, '`--`' and '`---`' are converted into dashes, and '`...`' becomes a compact set of dots.

11.5 Embedded LaTeX

Plain ASCII is normally 通常は sufficient for almost all note taking. Exceptions 例外には include 含まれます scientific notes, which これは often しばしば require 必用です mathematical symbols 数学のシンボルと and the occasional formula. たまに数式がLaTeX[3] が is widely used to typeset タイプセットするために scientific documents. Org-mode は supports サポートしています embedding LaTeX code into its files, because many academics are used to writing and reading LaTeX source code, and because it can be readily processed to produce pretty output for a number of export back-ends.

11.5.1 LaTeX fragments

Org-mode can contain LaTeX math fragments, and it それは supports サポートしています ways 方法を to process 処理する these これらを for several export back-ends. 複数のエクスポートバックエンドに対して When exporting to LaTeX へエクスポートするときには、the code このコードは is left as it is. そのままにされます When exporting to HTML へエクスポートするときには Org Org-mode は can use 使うことか either MathJax (`http://www.mathjax.org`)（節 12.9.10 [Math formatting in HTML export], ページ 164, を見てください）か or transcode 変換することができます the math into images 数式をイメージへ（節 11.5.2 [Previewing LaTeX fragments], ページ 140, を見てください）。

LaTeX fragments do not need any special marking at all. The following snippets are identified as LaTeX source code:

- Environments of any kind[4]. The only requirement is that the '`\begin`' statement appears on a new line, preceded by only whitespace.
- Text within the usual LaTeX math delimiters. To avoid 避けるため conflicts with currency specifications, single '`$`' characters are only recognized as math delimiters if the enclosed text contains at most two line breaks, is directly attached to the '`$`' characters with no whitespace in between, and if the closing '`$`' is followed by whitespace, punctuation or a dash. For the other delimiters, there is no such restriction, so when in doubt, use '`\(...\)`' as inline math delimiters.

例です:

```
\begin{equation}                          % arbitrary environments,
x=\sqrt{b}                                % even tables, figures
\end{equation}                            % etc

If $a^2=b$ and \( b=2 \), then the solution must be
either $$ a=+\sqrt{2} $$ or \[ a=-\sqrt{2} \].
```

[2] This behavior can be disabled with '`-`' export setting（節 12.2 [Export Settings], ページ 146, を見てください）

[3] LaTeX is a macro system based on Donald E. Knuth's TeX system. Many of the features described here as "LaTeX" are really from TeX, but for simplicity I am blurring this distinction.

[4] When MathJax is used, only the environments recognized by MathJax are processed. When dvipng, dvisvgm, or ImageMagick suite is used to create images, any LaTeX environment is handled.

LaTeX processing can be configured 設定することができます with the variable 変数`org-export-with-latex` を使って The default setting is デフォルト設定は`t` でありwhich これは means 意味します MathJax for HTML, and no processing for ASCII and LaTeX back-ends. You can also set this variable on a per-file basis ファイルごとに using 使って one of these lines:

'`#+OPTIONS: tex:t`'	Do the right thing automatically (MathJax)
'`#+OPTIONS: tex:nil`'	Do not process LaTeX fragments at all
'`#+OPTIONS: tex:verbatim`'	Verbatim export, for jsMath or so

11.5.2 Previewing LaTeX fragments

If you have a working LaTeX installation and '`dvipng`', '`dvisvgm`' or '`convert`' installed[5], LaTeX fragments can be processed to produce images of the typeset expressions to be used for inclusion while exporting to HTML（節 11.5.1 [LaTeX fragments], ページ 139, を見てください）、or for inline previewing within Org-mode.

You can customize カスタマイズすることができます the 変数`org-format-latex-options` と`org-format-latex-header` を to influence 影響を与えるために some aspects いくつかの側面に of the preview. プレビューの In particular, the `:scale` (and for HTML export, `:html-scale`) property of the former 前者お can be used 使うことができます to adjust 調節するために、the size of the preview images. プレビューイメージのサイズを

C-c C-x C-l (`org-toggle-latex-fragment`)
: Produce a preview image of the LaTeX fragment at point and overlay it over the source code. If there is no fragment at point, process all fragments in the current entry (between two headlines). When called with a prefix argument, 1 つの前置引数といっしょに呼ばれるときには、process the entire subtree. When called 呼ばれるときには with two prefix arguments, or when point is before the first headline, process the entire buffer.

You can turn on the previewing of all LaTeX fragments in a file with

```
#+STARTUP: latexpreview
```

To disable it, それを無効にするには、simply use 単純にこれを使ってください:

```
#+STARTUP: nolatexpreview
```

11.5.3 Using CDLaTeX to enter math

CDLaTeX mode is a minor mode マイナーモードです that is normally 通常は used 使われるin combination 組み合わせて with a major LaTeX mode like AUCTeX in order to speed- up insertion 挿入を of environments and math templates. Inside Org-mode, you can make use 使用することができま of some いくつかを of the features of CDLaTeX mode. You need to install インストールする必用があります'`cdlatex.el`' and '`texmathp.el`' (the latter comes also with AUCTeX) from `http://www.astro.uva.nl/~dominik/Tools/cdlatex`. Do not use CDLaTeX mode itself under Org-mode, but use the light version `org-cdlatex-mode` that comes as part of Org-mode. Turn it on for the current buffer with *M-x org-cdlatex-mode*, or for all Org files with

```
(add-hook 'org-mode-hook 'turn-on-org-cdlatex)
```

[5] These are respectively available at `http://sourceforge.net/projects/dvipng/`, `http://dvisvgm.bplaced.net/` and from the ImageMagick suite. Choose the converter by setting 設定することによってthe variable 変数`org-preview-latex-default-process` accordingly.

When this mode is enabled, the following features are present (for more details see the documentation of CDLaTeX mode):

C-c {

Insert 挿入します。an environment template. 環境のテンプレートを

TAB

The TAB key expands the template if point is inside a LaTeX fragment[6]. For example, 例えばTAB は expands 'fr' to '\frac{}{}' and position point correctly inside the first brace. Another TAB gets you into the second brace.

Even outside fragments, TAB expands environment abbreviations at the beginning of a line. For example, 例えば if you write 'equ' at the beginning of a line and press TAB, this abbreviation is expanded to an 'equation' environment. To get 得るには a list of all abbreviations, type *M-x cdlatex-command-help* とタイプしてください

^

_

Pressing _ and ^ inside a LaTeX fragment inserts 挿入します these characters これらの文字を together with a pair of braces. If you use TAB to move out of the braces, and if the braces surround only a single character or macro, they are removed again (depending 依存して on the 変数cdlatex-simplify-sub-super-scripts).

`

Pressing the backquote followed by a character inserts 挿入します math macros, also outside LaTeX fragments. If you wait more than 1.5 seconds after the backquote, a help window pops up.

'

Pressing the single-quote followed by another character modifies the symbol before point with an accent or a font. If you wait more than 1.5 seconds after the single-quote, a help window pops up. Character modification works only inside LaTeX fragments; outside the quote is normal.

11.6 Literal Examples

You can include literal examples that should not be subjected to markup. Such examples are typeset in monospace, so this is well suited for source code and similar examples.

```
#+BEGIN_EXAMPLE
  Some example from a text file.
#+END_EXAMPLE
```

Note 注意してください that such blocks may be *indented* in order to align nicely with indented text and in particular with plain list structure（節 2.7 [Plain Lists], ページ 13, を見てください）。For simplicity 単純なために when using 使うときには small examples, 小

[6] Org-mode has a method to test if point is inside such a fragment, see the documentation of the function org-inside-LaTeX-fragment-p.

さな例を you can also start はじめることもできます the example lines 例の行を with a colon followed by a space. There may also be additional whitespace before the colon:

```
Here is an example
   : Some example from a text file.
```

If the example is source code from a programming language, or any other text that can be marked up by Font Lock in Emacs, you can ask for the example to look like the fontified Emacs buffer[7]. This is done with the code block, where そこでは you also need to specify 指定する必用もあります the name of the major mode that should be used to fontify the example[8], see 節 15.2 [Structure Templates], ページ 240, for shortcuts to easily insert code blocks.

```
#+BEGIN_SRC emacs-lisp
  (defun org-xor (a b)
    "Exclusive or."
    (if a (not b) b))
 #+END_SRC
```

Both in '`example`' and in '`src`' snippets, you can add 追加することができます a '`-n`' スイッチを switch to the end of the '`#+BEGIN`' line, to get the lines of the example numbered. The =-n = takes an optional numeric argument specifying 指定している the starting line number of the block. If you use a '`+n`' switch, the numbering from the previous numbered snippet is continued in the current one. The '`+n`' switch can also take a numeric argument. This adds the value of the argument to the last line of the previous block to determine the starting line number.

```
#+BEGIN_SRC emacs-lisp -n 20
  ;; This exports with line number 20.
  (message "This is line 21")
#+END_SRC

#+BEGIN_SRC emacs-lisp +n 10
  ;; This is listed as line 31.
  (message "This is line 32")
#+END_SRC
```

In literal examples, Org interprets strings like '`(ref:name)`' as labels, and use them as targets for special hyperlinks like '`[[(name)]]`' ---i.e., すなわち the reference name enclosed in single parenthesis. In HTML, hovering the mouse over such a link remote-highlights the corresponding code line, which これが is kind of cool. 少しクールです

---

[7] This これは works 動きます automatically 自動的に for the HTML backend (it それには requires 必用です version 1.34 of the '`htmlize.el`' package, which これお you need to install). あなたがインストールする必用があります Fontified code chunks in LaTeX can be achieved 達成することができます using either the listings (`https://www.ctan.org/pkg/listings`) パッケージかminted (`https://www.ctan.org/pkg/minted`) パッケージを使って Refer to `org-export-latex-listings` を参照してください。for details. 詳細については

[8] Source code in code blocks may also be evaluated 評価することもできまますう either interactively インタラクティブにか or on export. エクスポート時に See 章 14 [Working with Source Code], ページ 211, を見てください for more information on evaluating code blocks.

You can also add a '`-r`' switch which *removes* the labels from the source code[9]. With the '`-n`' switch, links to these references are labeled by the line numbers from the code listing. Otherwise そうでなければ links リンクは use 使います the labels ラベルを with no parentheses. 括弧のないこれは例です:

```
#+BEGIN_SRC emacs-lisp -n -r
  (save-excursion                  (ref:sc)
     (goto-char (point-min))       (ref:jump)
#+END_SRC
In line [[(sc)]] we remember the current position. [[(jump)][Line (jump)]]
jumps to point-min.
```

Finally, 最後に、you can use '`-i`' を使うことができます to preserve 保存するために、the indentation of a specific code block 特定のコードブロックのインデントを（節 14.9 [Editing Source Code], ページ 233, を見てください）。

If the syntax for the label format conflicts with the language syntax, use a '`-l`' switch to change the format, for example 例えば

```
#+BEGIN_SRC pascal -n -r -l "((%s))"
```

変数`org-coderef-label-format` も見てください。

HTML export also allows examples to be published as text areas（節 12.9.11 [Text areas in HTML export], ページ 165, を見てください）。

Because the '`#+BEGIN`' ... '`#+END`' patterns need to be added so often, a shortcut is provided（節 15.2 [Structure Templates], ページ 240, を見てください）。

C-c ' (`org-edit-special`)
: Edit the source code example at point in its native mode. This works by switching to a temporary buffer with the source code. You need to exit by pressing *C-c '* を押すことににによって again[10]. The edited version then replaces the old version in the Org buffer. Fixed-width regions---where each line starts with a colon followed by a space---are edited using `artist-mode`[11] to allow creating ASCII drawings easily. Using this command in an empty line creates 作成します a new fixed-width region.

Calling `org-store-link`（節 4.5 [Handling Links], ページ 43, を見てください）while editing a source code example in a temporary buffer created with *C-c '* prompts プロンプトを出します for a label. Make sure that it is unique in the current buffer, and insert it with the proper formatting like '`(ref:label)`' at the end of the current line. Then the label is stored as a link '`(label)`', for retrieval with *C-c C-l*.

---

[9] Adding '`-k`' to '`-n -r`' *keeps* the labels in the source code while using line numbers for the links, which これが might be useful to 役に立つかもしれません explain those in an Org-mode example code.

[10] Upon exit, lines starting with '`*`', '`,*`', '`#+`' and '`,#+`' get a comma prepended, to keep them from being interpreted by Org as outline nodes or special syntax. These commas are stripped when editing with *C-c '*, and also before export.

[11] You may select 選択することができま a different-mode with the variable 変数`org-edit-fixed-width-region-mode` を使って

11.7 Images

An image is a link to an image file[12] that does not have a description part, for example 例えば

```
./img/cat.jpg
```

If you wish to define a caption for the image（節 11.8 [Captions], ページ 144, を見てください）and maybe a label for internal cross references（節 4.2 [Internal Links], ページ 40, を見てください）、make sure that the link is on a line by itself and precede it with '`CAPTION`' and '`NAME`' keywords as follows:

```
#+CAPTION: This is the caption for the next figure link (or table)
#+NAME:   fig:SED-HR4049
[[./img/a.jpg]]
```

Such images can be displayed within the buffer with the following command:

C-c C-x C-v (`org-toggle-inline-images`)

Toggle トグルします the inline display of linked images. When called with a prefix argument, 1 つの前置引数といっしょに呼ばれるときには also display images that do have a link description. You can ask for inline images to be displayed at startup by configuring 設定することによって the variable 変数`org-startup-with-inline-images` を[13].

11.8 Captions

You can assign a caption to a specific part of a document by inserting a '`CAPTION`' keyword immediately before it:

```
#+CAPTION: This is the caption for the next table (or link)
| ... | ... |
|-----+-----|
```

Optionally, the caption can take the form:

```
#+CAPTION[Short caption]: Longer caption.
```

Even though images and tables are prominent examples of captioned structures, the same caption mechanism can apply to many others ---e.g., 例えばLaTeX equations, source code blocks. Depending on 依存して the export back-end, エクスポートバックエンドに those これらは may or may not be handled.

11.9 Horizontal Rules

A line consisting of only dashes, and at least 5 of them, is exported as a horizontal line.

---

[12] What Emacs considers to be an image depends on `image-file-name-extensions` and `image-file-name-regexps`.

[13] The variable 変数`org-startup-with-inline-images` can be set within a buffer with the '`STARTUP`' options '`inlineimages`' and '`noinlineimages`'.

12 Exporting

At some point you might want to print your notes, publish them on the web, or share them with people not using Org. Org can convert and export documents to a variety of other formats while retaining as much structure（章 2 [Document Structure], ページ 6, を見てください）and markup（章 11 [Markup for Rich Contents], ページ 137, を見てください）as possible. 可能なかぎり

The libraries ライブラリは responsible for translating Org files to other formats are called *back-ends*. Org ships with support for the following back-ends:

- *ascii* (ASCII format)
- *beamer* (LaTeX Beamer format)
- *html* (HTML format)
- *icalendar* (iCalendar format)
- *latex* (LaTeX format)
- *md* (Markdown format)
- *odt* (OpenDocument Text format)
- *org* (Org format)
- *texinfo* (Texinfo format)
- *man* (Man page format)

Users can install libraries for additional formats from the Emacs packaging system. For easy discovery, these packages have a common naming scheme: `ox-NAME`, where *NAME* is a format. For example, `ox-koma-letter` for *koma-letter* back-end. More libraries can be found in the '`contrib/`' directory（節 1.2 [Installation], ページ 2, を見てください）。

Org only loads back-ends for the following formats by default: デフォルトで ASCII, HTML, iCalendar, LaTeX, and ODT. Additional back-ends can be loaded in either of two ways: by configuring 設定することによって the `org-export-backends` variable, or by requiring libraries in the Emacs init file. For example, to load the Markdown back-end, add this to your Emacs config:

```
(require 'ox-md)
```

12.1 The Export Dispatcher

The export dispatcher is the main interface for Org's exports. A hierarchical menu presents the currently configured export formats. Options are shown as easy toggle switches on the same screen.

Org also has a minimal prompt interface for the export dispatcher. When the variable `org-export-dispatch-use-expert-ui` is set 設定されているときには to a non-`nil` value, `nil` でない値に Org prompts プロンプトを出します in the minibuffer. To switch 切り替えるには back to the hierarchical menu, press `?` を押してください

`C-c C-e` (`org-export`)
: Invokes the export dispatcher interface. The options show default settings. The `C-u` prefix argument preserves options from the previous export, including 含む any sub-tree selections.

Org exports the entire buffer by default. デフォルトで If the Org buffer has あれば an active region, アクティブなリージョンが then Org exports just that region.

Within the dispatcher interface, the following key combinations can further alter what is exported, and how.

`C-a`

Toggle トグルします asynchronous export. 非同期エクスポートを Asynchronous export uses 使います an external Emacs process with a specially configured initialization file to complete the exporting process in the background, バックグラウンドで without tying-up Emacs. Emacs を占有することなく This is particularly useful これが特に役に立ちます。when exporting long documents. 長いドキュメントをエクスポートするときに、

Output from an asynchronous export is saved on the *export stack*. To view 見るには this stack, このスタックを call the export dispatcher with a double `C-u` prefix argument. If already in the export dispatcher menu, `&` が displays 表示します the stack. このスタックを

You can make asynchronous export the default デフォルトにすることもできんます。by setting `org-export-in-background` を設定することによって

You can set 設定することげんえきます the initialization file 初期化ファイルを used 使われる by the background process by setting `org-export-async-init-file` を設定することによって

`C-b`

Toggle トグルします body-only export. Useful 役に立ちます for excluding headers and footers in the export. Affects only those back-end formats that have sections like '`<head>...</head>`' in HTML.

`C-s`

Toggle トグルします sub-tree export. When turned on, Org exports only the sub-tree starting from point position at the time the export dispatcher was invoked. Org uses 使います the top heading of this sub-tree as the document's title. If point ポイントが is not on a heading, 見出しの上にんあいときには Org Org-mode は uses 使います the nearest enclosing header. If point ポイントが is in the document preamble, Org signals an error and aborts export.

To make sub-tree export the default, デフォルトにするには customize カスタマイズしてください the variable 変数`org-export-initial-scope` を

`C-v`

Toggle トグルします visible-only export. This is useful これが役に立ちます for exporting only certain parts of an Org document by adjusting the visibility of particular headings.

12.2 Export Settings

Export options エクスポートオプションを can be set: 設定することができます globally グローバルい with variables; for an individual file by making variables buffer-local with in-buffer settings（節 15.7 [In-buffer Settings], ページ 243, を見てください）; by setting

設定することによって individual keywords or specifying 指定することいよって them in compact form with the '`OPTIONS`' keyword; or for a tree by setting 設定することによって properties プロパティを（章 7 [Properties and Columns], ページ 66, を見てください）。Options set at a specific level override options set at a more general level.

In-buffer settings may appear anywhere どこにでも in the file, either directly or indirectly 間接的に through a file included using '`#+SETUPFILE: filename or URL`' syntax. Option keyword sets tailored to a particular back-end can be inserted from the export dispatcher（節 12.1 [The Export Dispatcher], ページ 145, を見てください）using 使って the '`Insert template`' command by pressing `#` を押すことににによって To insert 挿入するには keywords individually, a good way to make sure the keyword キーワードが is correct 正しいことを is to type '`#+`' と and then to use 使うということです*`M-TAB`* を

for completion. 補完のために

The export keywords available for every back-end, and their equivalent global variables, include:

'`AUTHOR`' ドキュメントの著者です(`user-full-name`)。

'`CREATOR`' Entity responsible for output generation (`org-export-creator-string`).

'`DATE`' A date or a time-stamp[1].

'`EMAIL`' The email address (`user-mail-address`).

'`LANGUAGE`'
Language to use for translating certain strings (`org-export-default-language`). With '`#+LANGUAGE: fr`', for example, 例えば Org translates '`Table of contents`' to the French '`Table des matières`'[2].

'`SELECT_TAGS`'
デフォルト値は'`("export")`' です。When a tree is tagged with '`export`' (`org-export-select-tags`), Org selects that tree and its sub-trees for export. Org excludes trees with '`noexport`' tags, see below. When selectively exporting files with '`export`' tags set, Org does not export any text that appears before the first headline.

'`EXCLUDE_TAGS`'
デフォルト値は'`("noexport")`' です。When a tree is tagged with '`noexport`' (`org-export-exclude-tags`), Org excludes that tree and its sub-trees from export. Entries tagged with '`noexport`' are unconditionally excluded from the export, even if they have an '`export`' tag. Even if a sub-tree is not exported, Org executes any code blocks contained there.

'`TITLE`' Org displays 表示します this title. For long titles, use multiple '`#+TITLE`' lines.

---

[1] The variable 変数`org-export-date-timestamp-format` defines 定義します how 方法を this timestamp a re exported.

[2] For export to LaTeX format---or LaTeX-related formats such as Beamer—, the '`org-latex-package-alist`' 変数を needs 必用です further configuration. 節 12.10.2 [LaTeX specific export settings], ページ 168, を見てください。

'EXPORT_FILE_NAME'
: The name of the output file to be generated. Otherwise, そうでなければ Org Org-mode は generates 生成します the file name based on the buffer name and the extension based on the back-end format.

The 'OPTIONS' keyword is a compact form. To configure 設定するには multiple options, 複数のオプションを use 使ってください several 'OPTIONS' lines. 'OPTIONS' recognizes the following arguments.

'
: Toggle トグルします smart quotes (org-export-with-smart-quotes). Depending 依存して on the language used, 使われている言語に when activated, Org treats pairs of double quotes as primary quotes, pairs of single quotes as secondary quotes, and single quote marks as apostrophes.

*
: Toggle トグルします emphasized text (org-export-with-emphasize).

-
: Toggle トグルします conversion of special strings (org-export-with-special-strings).

:
: Toggle トグルします fixed-width sections (org-export-with-fixed-width).

<
: Toggle トグルします inclusion 含めるかどうかを of time/date active/inactive stamps (org-export-with-timestamps).

\n
: Toggles トグルします whether to preserve line breaks (org-export-preserve-breaks).

^
: Toggle トグルしますTEX-like syntax for sub- and superscripts. If you write '^:{}', 'a_{b}' is interpreted, but the simple 'a_b' is left as it is (org-export-with-sub-superscripts).

arch
: Configure 設定します how archived trees are exported. When set to headline に設定されているときには、the export process skips the contents and processes only the headlines (org-export-with-archived-trees).

author
: Toggle トグルします inclusion 含めるかどうかを of author name into exported file (org-export-with-author).

broken-links
: Toggles トグルします if Org should continue exporting upon finding a broken internal link. When set to mark に設定されているときには、Org clearly marks the problem link in the output (org-export-with-broken-links).

c
: Toggle トグルします inclusion 含めるかどうかを of CLOCK keywords (org-export-with-clocks).

creator
: Toggle トグルします inclusion 含めるかどうかを of creator information in the exported file (org-export-with-creator).

d
: Toggles トグルします inclusion 含めるかどうかを of drawers, or list of drawers to include, or list of drawers to exclude (org-export-with-drawers).

date
: Toggle トグルします inclusion 含めるかどうかを of a date into exported file (org-export-with-date).

`e` Toggle トグルします inclusion 含めるかどうかを of entities エンティティを(`org-export-with-entities`).

`email` Toggle トグルします inclusion 含めるかどうかを of the author's e-mail into exported file (`org-export-with-email`).

`f` Toggle トグルします the inclusion 含めるかどうかを of footnotes 脚注を(`org-export-with-footnotes`).

`H` Set the number of headline levels for export (`org-export-headline-levels`). Below that level, headlines are treated differently. In most back-ends, they become list items.

`inline` Toggle トグルします inclusion 含めるかどうかを of inlinetasks インラインタスクを(`org-export-with-inlinetasks`).

`num` Toggle トグルします section-numbers (`org-export-with-section-numbers`). When set to number N, 数値の N に設定されているときには、Org numbers only those headlines at level N or above. Set '`UNNUMBERED`' プロパティを to non-`nil` `nil` でない値に to disable 無効にするには numbering of heading and subheadings entirely. Moreover, when the value is '`notoc`' the headline, and all its children, do not appear in the table of contents either（節 12.3 [Table of Contents], ページ 150, を見てください）

`p` Toggle トグルします export of planning information (`org-export-with-planning`). `"Planning information"` comes from lines located right after the headline and contain any combination of these cookies: '`SCHEDULED`', '`DEADLINE`', or '`CLOSED`'.

`pri` Toggle トグルします inclusion 含めるかどうかを of priority cookies 優先順位のクッキーを(`org-export-with-priority`).

`prop` Toggle トグルします inclusion 含めるかどうかを of property drawers, プロパティドロワーを or list the properties to include (`org-export-with-properties`).

`stat` Toggle トグルします inclusion 含めるかどうかを of statistics cookies 統計のクッキーを(`org-export-with-statistics-cookies`).

`tags` Toggle トグルします inclusion 含めるかどうかを of tags, タグを may also be `not-in-toc` (`org-export-with-tags`).

`tasks` Toggle トグルします inclusion 含めるかどうかを of tasks タスクを(TODO items); or `nil` to remove all tasks; or `todo` to remove DONE tasks; or list the keywords to keep (`org-export-with-tasks`).

`tex` `nil` does not export; `t` exports; `verbatim` keeps everything in verbatim (`org-export-with-latex`).

`timestamp` Toggle トグルします inclusion 含めるかどうかを of the creation time 作成時間を in the exported file (`org-export-time-stamp-file`).

`title` Toggle トグルします inclusion 含めるかどうかを of title タイトルを(`org-export-with-title`).

`toc` Toggle トグルします inclusion 含めるかどうかを of the table of contents, 目次を or set the level limit (`org-export-with-toc`).

`todo` Toggle トグルします inclusion of TODO keywords into exported text (`org-export-with-todo-keywords`).

`|` Toggle トグルします inclusion 含めるかどうかを of tables 表を(`org-export-with-tables`).

When exporting エクスポートするときに sub-trees, サブツリーを special node properties 特別なノードごとのプロパティが can override 上書きすることができます the above keywords. 上のキーワードを These properties have an '`EXPORT_`' prefix. For example, 例えば'`DATE`' becomes, '`EXPORT_DATE`' when used 使われるときには for a specific sub-tree. Except 以外は for '`SETUPFILE`', all other keywords listed above have 持ちます an '`EXPORT_`' equivalent.

If `org-export-allow-bind-keywords` が`nil` でない値であれば、Emacs variables can become buffer-local during export by using 使うことによって the '`BIND`' キーワードを Its syntax その文法は is '`#+BIND: variable value`' です This これが is particularly useful 特に役に立ちます for in-buffer settings that cannot be changed 変更することができない using keywords. キーワードを使って

12.3 Table of Contents

The table of contents 目次は includes all headlines 全ての見出しを含んでいます。in the document. ドキュメントの中の Its depth is therefore よってその深さは the same 同じです as the headline levels 見出しのレベルと in the file. ファイルの中の If you need to use 使う必用があれば a different depth, 異なる深さを or turn it それを off entirely, 完全にオフにするには set 設定してください the `org-export-with-toc` 変数を accordingly. 適切に You can achieve 達成することができます the same 同じことを on a per file basis, ファイルごとに using 使って the following '`toc`' item in '`OPTIONS`' keyword:

```
#+OPTIONS: toc:2          (only include two levels in TOC)
#+OPTIONS: toc:nil        (no default TOC at all)
```

Org includes both numbered and unnumbered headlines in the table of contents

12.4 Include Files

During export, you can include the content of another file. For example, to include your '`.emacs`' file, you could use:

```
#+INCLUDE: "~/.emacs" src emacs-lisp
```

The first parameter is the file name to include. The optional second parameter specifies 指定します the block type: '`example`', '`export`' or '`src`'. The optional third parameter specifies 指定します the source code language to use for formatting the contents. This is relevant to both '`export`' and '`src`' block types.

If an included file is specified as having a markup language, Org neither checks for valid syntax nor changes the contents in any way. For example 例と and source blocks, Org code-escapes the contents before inclusion.

If an included file is not specified as having any markup language, Org assumes it be in Org format and proceeds as usual with a few exceptions. Org makes the footnote labels

（節 2.10 [Creating Footnotes], ページ 17, を見てください）in the included file local to that file. The contents of the included file belong to the same structure ---headline, item ---containing 含んでいる the '`INCLUDE`' keyword. In particular, headlines within the file become children of the current section. That behavior そのふるまいを can be changed 変更することができます by providing 提供することによって an additional keyword parameter, '`:minlevel`'. It shifts the headlines in the included file to become the lowest level. For example, 例えば this syntax makes the included file a sibling of the current top-level headline:

```
#+INCLUDE: "~/my-book/chapter2.org" :minlevel 1
```

Inclusion of only portions of files are specified using ranges parameter with '`:lines`' keyword. The line at the upper end of the range will not be included. The start and/or the end of the range may be omitted to use the obvious defaults.

'`#+INCLUDE: "~/.emacs" :lines "5-10"`'	Include lines 5 to 10, 10 excluded
'`#+INCLUDE: "~/.emacs" :lines "-10"`'	Include lines 1 to 10, 10 excluded
'`#+INCLUDE: "~/.emacs" :lines "10-"`'	Include lines from 10 to EOF

Inclusions may specify a file-link to extract an object matched by `org-link-search`[3]（節 4.8 [Search Options], ページ 48, を見てください）。The ranges for '`:lines`' keyword are relative to the requested element. Therefore,

```
#+INCLUDE: "./paper.org::*conclusion" :lines 1-20
```

includes the first 20 lines of the headline named '`conclusion`'.

To extract 抽出するいは only the contents 中身だけを of the matched object, マッチしたオブジェクトの set 設定してください。'`:only-contents`' プロパティを to non-`nil`. `nil` でない値に This omits 省略します any planning lines or property drawers. For example, 例えば to include 含めるには the body of the heading with the custom ID '`theory`', you can use 使うことができます

```
#+INCLUDE: "./paper.org::#theory" :only-contents t
```

The following command allows navigating back and forth to the included document:

C-c ' (`org-edit~special`)
: Visit the included file at point.

12.5 Macro Replacement

Macros replace text snippets during export. Macros are defined globally in `org-export-global-macros`, or document-wise with the following syntax:

```
#+MACRO: name   replacement text; $1, $2 are arguments
```

which can be referenced using '`{{{name(arg1, arg2)}}}`'[4]. For example

```
#+MACRO: poem The rose is $1, The violet's $2. Life's ordered: Org assists you.
{{{poem(red,blue)}}}
```

---

[3] Note 注意してください that `org-link-search-must-match-exact-headline` is locally bound to non-`nil`. `nil` でない値に Therefore, `org-link-search` only matches headlines and named elements.

[4] Since commas separate the arguments, commas within arguments have to be escaped with the backslash character. So only those backslash characters before a comma need escaping with another backslash character.

becomes

```
The rose is red, The violet's blue. Life's ordered: Org assists you.
```

As a special case, Org parses any replacement text starting with '`(eval`' as an Emacs Lisp expression and evaluates it accordingly. Within such templates, arguments become strings. Thus, よって the following macro

```
#+MACRO: gnucheck (eval (concat "GNU/" (capitalize $1)))
```

turns '`{{{gnucheck(linux)}}}`' into '`GNU/Linux`' during export.

Org recognizes macro references in following Org markup areas: paragraphs, headlines, verse blocks, tables cells and lists. Org also recognizes macro references in keywords, such as '`CAPTION`', '`TITLE`', '`AUTHOR`', '`DATE`', and for some back-end specific export options.

Org comes with following pre-defined macros:

'`{{{keyword(NAME)}}}`'
'`{{{title}}}`'
'`{{{author}}}`'
'`{{{email}}}`'

The '`keyword`' macro collects all values from *NAME* keywords throughout the buffer, separated with white space. '`title`', '`author`' and '`email`' macros are shortcuts for, respectively, '`{{{keyword(TITLE)}}}`', '`{{{keyword(AUTHOR)}}}`' and '`{{{keyword(EMAIL)}}}`'.

'`{{{date}}}`'
'`{{{date(FORMAT)}}}`'

This macro refers to the '`DATE`' keyword. *FORMAT* is an optional argument to the '`date`' macro that is used only if '`DATE`' is a single timestamp. *FORMAT* should be a format string understood by `format-time-string`.

'`{{{time(FORMAT)}}}`'
'`{{{modification-time(FORMAT, VC)}}}`'

These macros refer to the document's date and time of export and date and time of modification. *FORMAT* is a string understood by `format-time-string` いよって If the second argument to the `modification-time` macro is non-`nil`, `nil` でない値の場合には Org uses 使います'`vc.el`' を to retrieve the document's modification time from the version control system. Otherwise そうでなければ Org Org-mode は reads 読みます the file attributes.

'`{{{input-file}}}`'

This macro refers to the filename of the exported file.

'`{{{property(PROPERTY-NAME)}}}`'
'`{{{property(PROPERTY-NAME, SEARCH OPTION)}}}`'

This macro returns the value of property *PROPERTY-NAME* in the current entry. If *SEARCH-OPTION*（節 4.8 [Search Options], ページ 48, を見てください）refers to a remote entry, use it それを instead. かわりに

'`{{{n}}}`'
'`{{{n(NAME)}}}`'
'`{{{n(NAME, ACTION)}}}`'

This macro implements custom counters by returning the number of times the macro has been expanded so far while exporting the buffer. You can create

more than one counter using different *NAME* values. If *ACTION* is '`-`', previous value of the counter is held, i.e., すなわち the specified counter is not incremented. If the value is a number, the specified counter is set to that value. If it is any other non-empty string, the specified counter is reset to 1. You may leave *NAME* empty to reset the default counter.

Moreover, inline source blocks（節 14.1 [Structure of Code Blocks], ページ 212, を見てください）use the special '`results`' macro to mark their output. As such, you are advised against re-defining it, unless you know what you are doing.

The surrounding brackets can be made invisible by setting 設定することによって`org-hide-macro-markers` を to a non-`nil` value. `nil` でない値に

Org expands macros at the very beginning of the export process.

12.6 Comment Lines

Lines starting with zero or more whitespace characters followed by one '`#`' and a whitespace are treated as comments and, as such, are not exported.

Likewise, regions surrounded by '`#+BEGIN_COMMENT`' ... '`#+END_COMMENT`' are not exported.

Finally, 最後に a '`COMMENT`' keyword at the beginning of an entry, but after any other keyword or priority cookie, comments out the entire subtree. In this case, この場合には the subtree is not exported エクスポートされず and no code block within it is executed either

12.7 ASCII/Latin-1/UTF-8 export

ASCII export produces an output file containing 含んでいる only plain ASCII characters. This is the simplest and most direct text output. It does not contain any Org markup. Latin-1 and UTF-8 export use additional characters and symbols available in these encoding standards. All three of these export formats offer the most basic of text output for maximum portability.

On export, Org fills and justifies text according 従って to the text width set in `org-ascii-text-width` の中で

Org exports links using a footnote-like style where the descriptive part is in the text and the link is in a note before the next heading. 詳細については変数`org-ascii-links-to-notes` を見てください。

ASCII export commands

C-c C-e t a (`org-ascii-export-to-ascii`)
C-c C-e t l
C-c C-e t u

Export as an ASCII ファイルとしてエクスポートします。with a '`.txt`' という拡張子をもつ For '`myfile.org`' に対して、Org Org-mode は exports to '`myfile.txt`' へエクスポートし、overwriting without warning. 警告なしで上書きします。For '`myfile.txt`', Org exports エクスポートします to '`myfile.txt.txt`' in order to prevent data loss. データのロスを防ぐために

C-c C-e t A (org-ascii-export-to-ascii)
C-c C-e t L
C-c C-e t U
: Export エクスポートします to a temporary buffer. 一時的なバッファへ Does not create a file. ファイルを作成しません。

ASCII specific export settings

The ASCII エクスポートバックエンドには has あります one extra keyword for customizing ASCII output. Setting this keyword このキーワードの設定は works similar 同じように動きます to the general options 一般的なオプションと（節 12.2 [Export Settings], ページ 146, を見てください）.

'SUBTITLE'
: The document subtitle. For long subtitles, use multiple '#+SUBTITLE' lines in the Org file. Org prints them on one continuous line, wrapping into multiple lines if necessary. 必用であれば

Header and sectioning structure

Org Org-mode は converts 変換します the first three outline levels 最初の 3 つのアウトラインのレベルを into headlines for ASCII export. 残りのレベルはリストに変えられます。To change 変更するには this cut-off point このカットオフポイントを where levels become lists, see 節 12.2 [Export Settings], ページ 146, を見てください。

Quoting ASCII text

To insert 挿入するには text ithin the Org file by the ASCII back-end, use one the following constructs, inline, keyword, or export block:

```
Inline text @@ascii:and additional text@@ within a paragraph.

#+ASCII: Some text

#+BEGIN_EXPORT ascii
Org exports text
in this block
only when using 使うときにだけ
ASCII バックエンドを
#+END_EXPORT
```

ASCII specific attributes

ASCII back-end recognizes only one attribute, ':width', which これは specifies 指定します the width 幅を of a horizontal rule in number of characters. The keyword and syntax for specifying widths is:

```
#+ATTR_ASCII: :width 10
-----
```

ASCII special blocks

Besides '#+BEGIN_CENTER' blocks（節 11.1 [Paragraphs], ページ 137, を見てください） ASCII back-end has these two left and right justification blocks:

```
#+BEGIN_JUSTIFYLEFT
It's just a jump to the left...
#+END_JUSTIFYLEFT

#+BEGIN_JUSTIFYRIGHT
...and then a step to the right.
#+END_JUSTIFYRIGHT
```

12.8 Beamer Export

Org uses 使います Beamer export to convert an Org file tree structure into high-quality interactive slides for presentations. Beamer is a LaTeX document class for creating presentations in PDF, HTML, and other popular display formats.

12.8.1 Beamer export commands

C-c C-e l b (`org-beamer-export-to-latex`)
: Export エクスポートします。as LaTeX ファイルとして with a '`.tex`' という拡張子をもつ For '`myfile.org`' に対して、Org Org-mode は exports to '`myfile.tex`' へエクスポートし、overwriting without warning. 警告なしで上書きします。

C-c C-e l B (`org-beamer-export-as-latex`)
: Export エクスポートします to a temporary buffer. 一時的なファイルへ Does not create a file. ファイルを作成しません。

C-c C-e l P (`org-beamer-export-to-pdf`)
: Export エクスポートします as LaTeX ファイルとして and then convert it to PDF format.

C-c C-e l O
: Export エクスポートします as LaTeX ファイルとして convert it to PDF format, and then open the PDF file.

12.8.2 Beamer specific export settings

Beamer エクスポートバックエンドには has あります several additional keywords 複数の追加のキーワードが for customizing カスタマイズするための Beamer output. Beamer 出力を These keywords work similar 同じように動きます to the general options settings（節 12.2 [Export Settings], ページ 146, を見てください）.

'`BEAMER_THEME`'
: The Beamer layout theme (`org-beamer-theme`). Use 使ってください square brackets for options. オプションに対しては大括弧を例です:

```
#+BEAMER_THEME: Rochester [height=20pt]
```

'`BEAMER_FONT_THEME`'
: The Beamer font theme.

'`BEAMER_INNER_THEME`'
: The Beamer inner theme.

'`BEAMER_OUTER_THEME`'
: The Beamer outer theme.

'BEAMER_HEADER'
: Arbitrary lines inserted in the preamble, just before the 'hyperref' settings.

'DESCRIPTION'
: The document description. For long descriptions, use multiple 'DESCRIPTION' keywords. By default, デフォルトで'hyperref' inserts 挿入します'DESCRIPTION' as metadata. Use 使ってくださいorg-latex-hyperref-template を to configure document metadata. Use 使ってくださいorg-latex-title-command を to configure 設定するには typesetting of description as part of front matter.

'KEYWORDS'
: The keywords for defining the contents of the document. Use 使ってください multiple 'KEYWORDS' lines if necessary. 必用であれば By default, デフォルトで'hyperref' は inserts 挿入します'KEYWORDS' を as metadata. メタデータとして Use 使ってくださいorg-latex-hyperref-template を to configure 設定するためには、document metadata. ドキュメントのメタデータを Use 使ってくださいorg-latex-title-command を to configure 設定するには typesetting of keywords as part of front matter.

'SUBTITLE'
: Document's subtitle. For typesetting, use org-beamer-subtitle-format string. Use 使ってくださいorg-latex-hyperref-template を to configure 設定するには document metadata. ドキュメントのメタデータを Use 使ってくださいorg-latex-title-command を to configure 設定するには typesetting of subtitle as part of front matter.

12.8.3 Frames and Blocks in Beamer

Org transforms heading levels into Beamer's sectioning elements, frames and blocks. Any Org tree with a not-too-deep-level nesting should in principle be exportable as a Beamer presentation.

- Org headlines become Beamer frames when the heading level in Org is equal to org-beamer-frame-level or 'H' value in a 'OPTIONS' line（節 12.2 [Export Settings], ページ 146, を見てください）.

  Org overrides headlines to frames conversion for the current tree of an Org file if it encounters the 'BEAMER_ENV' property set to 'frame' or 'fullframe'. Org ignores whatever org-beamer-frame-level happens to be for that headline level in the Org tree. In Beamer terminology, a full frame is a frame without its title.

- Org exports a Beamer frame's objects as block environments. Org can enforce wrapping in special block types when 'BEAMER_ENV' property is set[5]. For valid values see org-beamer-environments-default. To add more values, see org-beamer-environments-extra.

- If 'BEAMER_ENV' is set to 'appendix' に設定されていたら Org exports the entry as an appendix. When set to 'note' に設定されているときには、Org exports エクスポートします the entry as a note within the frame or between frames, depending on 依存して the entry's heading level. エントリーの見出しのレベルに When set to 'noteNH' に設

[5] If 'BEAMER_ENV' is set, Org export adds 'B_environment' tag to make it visible. The tag serves as a visual aid and has no semantic relevance.

定されているときには、Org exports エクスポートします the entry as a note without its title. When set to '`againframe`' に設定されているときには、Org exports エクスポートします the entry このエントリーを with '`\againframe`' command, which makes setting the '`BEAMER_REF`' property mandatory because '`\againframe`' needs 必用なので frame to resume.

When '`ignoreheading`' is set, Org export ignores the entry's headline but not its content. This is useful これが役に立ちます for inserting content between frames. It is also useful 役に立ちます for properly closing a '`column`' environment. @end itemize

When '`BEAMER_ACT`' is set for a headline, Org export translates that headline as an overlay or action specification. When enclosed in square brackets, Org export makes the overlay specification a default. Use 使ってください'`BEAMER_OPT`' を to set any options applicable to the current Beamer frame or block. The Beamer エクスポートバックエンドは wraps with appropriate angular or square brackets. It also adds the '`fragile`' option for any code that may require a verbatim block.

To create 作成するには a column コラムを on the Beamer slide, use 使ってください the '`BEAMER_COL`' property for its headline in the Org file. Set the value of '`BEAMER_COL`' to a decimal number representing the fraction of the total text width. Beamer エクスポートは uses 使います this value to set the column's width and fills the column with the contents of the Org entry. If the Org entry has no specific environment defined, Beamer export ignores the heading. If the Org entry has あれば a defined environment, Beamer export uses 使います the heading as title. Behind the scenes, Beamer export automatically 自動的に handles LaTeX column separations for contiguous headlines. To manually adjust them for any unique configurations needs, use the '`BEAMER_ENV`' プロパティを使ってください。

12.8.4 Beamer specific syntax

Since Org の Beamer エクスポートバックエンドが is an extension of the LaTeX back-end, it recognizes other LaTeX specific syntax ---for example, 例えば'`#+LATEX:`' or '`#+ATTR_LATEX:`'. 詳細については節 12.10 [LaTeX Export], ページ 167, を見てください。

Beamer export wraps the table of contents generated with '`toc:t`' '`OPTION`' keyword in a '`frame`' environment. Beamer export does not wrap the table of contents generated with '`TOC`' keyword（節 12.3 [Table of Contents], ページ 150, を見てください）. Use 使ってください square brackets 大括弧を for specifying options. オプションを指定するには

```
#+TOC: headlines [currentsection]
```

Insert 挿入します Beamer-specific code using the following constructs:

```
#+BEAMER: \pause

#+BEGIN_EXPORT beamer
  Only Beamer エクスポートバックエンドだきゃ
  exports エクスポートします
  this. これを
#+END_BEAMER

Text @@beamer:some code@@ within a paragraph.
```

Inline constructs, such as the last one above, are useful 役に立ちます for adding overlay specifications to objects with `bold`, `item`, `link`, `radio-target` and `target` types. Enclose the value in angular brackets and place the specification at the beginning of the object as shown in this example:

```
A *@@beamer:<2->@@useful* feature
```

Beamer export recognizes the '`ATTR_BEAMER`' keyword with the following attributes from Beamer configurations: '`:environment`' for changing local Beamer environment, '`:overlay`' for specifying Beamer overlays in angular or square brackets, and '`:options`' for inserting optional arguments.

```
#+ATTR_BEAMER: :environment nonindentlist
- item 1, not indented
- item 2, not indented
- item 3, not indented

#+ATTR_BEAMER: :overlay <+->
- item 1
- item 2

#+ATTR_BEAMER: :options [Lagrange]
Let $G$ be a finite group, and let $H$ be
a subgroup of $G$.  Then the order of $H$ divides the order of $G$.
```

12.8.5 Editing support

The `org-beamer-mode` is a special minor mode for faster editing of Beamer documents.

```
#+STARTUP: beamer
```

C-c C-b (`org-beamer-select-environment`)
: The `org-beamer-mode` は provides 提供しています this key for quicker selections in Beamer normal environments, and for selecting the '`BEAMER_COL`' property.

12.8.6 A Beamer example

Here これは is an example 例です of an Org document ready for Beamer export.

```
;#+TITLE: Example Presentation
;#+AUTHOR: Carsten Dominik
;#+OPTIONS: H:2 toc:t num:t
;#+LATEX_CLASS: beamer
;#+LATEX_CLASS_OPTIONS: [presentation]
;#+BEAMER_THEME: Madrid
;#+COLUMNS: %45ITEM %10BEAMER_ENV(Env) %10BEAMER_ACT(Act) %4BEAMER_COL(Col) %8BEAMER_O

* This is the first structural section

** Frame 1
*** Thanks to Eric Fraga                                           :B_block:
    :PROPERTIES:
    :BEAMER_COL: 0.48
    :BEAMER_ENV: block
    :END:
```

```
    for the first viable Beamer setup in Org
*** Thanks to everyone else                                   :B_block:
    :PROPERTIES:
    :BEAMER_COL: 0.48
    :BEAMER_ACT: <2->
    :BEAMER_ENV: block
    :END:
    for contributing to the discussion
**** This will be formatted as a beamer note                  :B_note:
     :PROPERTIES:
     :BEAMER_env: note
     :END:
** Frame 2 (where we will not use columns)
*** Request
    Please test this stuff!
```

12.9 HTML Export

Org-mode contains an HTML exporter with extensive HTML formatting compatible with XHTML 1.0 strict standard.

12.9.1 HTML export commands

C-c C-e h h (`org-html-export-to-html`)
: Export エクスポートします as HTML ファイルとして with a '`.html`' という拡張子をおつ For '`myfile.org`' に対して、Org Org-mode は exports エクスポートします to '`myfile.html`' へ overwriting without warning. 警告なしで上書きします。{{{kbd{C-c C-e h o)}}} は exports エクスポートします to HTML へ and opens 開きます。it in a web browser. それをウエブブラウザの中で

C-c C-e h H (`org-html-export-as-html`)
: Exports エクスポートします to a temporary buffer. 一時的なバッファへ Does not create a file. ファイルを作成しません。

12.9.2 HTML specific export settings

HTML エクスポートには has あります a number of keywords, たくさんのキーワードが similar 似ている to the general options settings 一般的なオプション設定と described in 節 12.2 [Export Settings], ページ 146, の中で説明されている

'`DESCRIPTION`'
: This is the document's description, which これを the HTML exporter inserts 挿入します it それを as a HTML meta tag in the HTML file. For long descriptions, use multiple '`DESCRIPTION`' lines. The exporter takes care of wrapping the lines properly.

'`HTML_DOCTYPE`'
: Specify the document type, for example: 例えば HTML5 (`org-html-doctype`).

'`HTML_CONTAINER`'
: Specify the HTML container, such as '`div`', for wrapping sections and elements (`org-html-container-element`).

'HTML_LINK_HOME'
: The URL for home link (`org-html-link-home`).

'HTML_LINK_UP'
: The URL for the up link of exported HTML pages (`org-html-link-up`).

'HTML_MATHJAX'
: Options for MathJax (`org-html-mathjax-options`). MathJax is used to typeset LaTeX math in HTML documents. See 節 12.9.10 [Math formatting in HTML export], ページ 164, を見てください。for an example. 例については

'HTML_HEAD'
: Arbitrary lines for appending to the HTML document's head (`org-html-head`).

'HTML_HEAD_EXTRA'
: More arbitrary lines for appending to the HTML document's head (`org-html-head-extra`).

'KEYWORDS'
: Keywords to describe the document's content. HTML exporter inserts 挿入します these keywords as HTML meta tags. For long keywords, use multiple 'KEYWORDS' lines.

'LATEX_HEADER'
: Arbitrary lines for appending to the preamble; HTML exporter appends when transcoding LaTeX fragments to images（節 12.9.10 [Math formatting in HTML export], ページ 164, を見てください）。

'SUBTITLE'
: The document's subtitle. HTML exporter formats subtitle if document type is 'HTML5' and the CSS has a 'subtitle' class.

Some of these keywords are explained in more detail in the following sections of the manual.

12.9.3 HTML doctypes

Org can export to various (X)HTML flavors.

Set the `org-html-doctype` variable for different (X)HTML variants. Depending 依存して on the variant, バリアントに the HTML exporter adjusts 調節します the syntax of HTML conversion accordingly. Org includes the following ready-made variants:

- `"html4-strict"`
- `"html4-transitional"`
- `"html4-frameset"`
- `"xhtml-strict"`
- `"xhtml-transitional"`
- `"xhtml-frameset"`
- `"xhtml-11"`
- `"html5"`
- `"xhtml5"`

詳細については変数`org-html-doctype-alist` を見てください。デフォルトは`"xhtml-strict"` です。

Org's HTML exporter does not by default デフォルトで enable new block elements introduced with the HTML5 standard. To enable them, set `org-html-html5-fancy` を to non-`nil` `nil` でない値に Or use an '`OPTIONS`' line in the file to set '`html5-fancy`'.

HTML5 documents can now have arbitrary '`#+BEGIN`' ... '`#+END`' blocks. 例です:

```
#+BEGIN_aside
  Lorem ipsum
#+END_aside
```

exports to: エクスポートされます:

```
<aside>
  <p>Lorem ipsum</p>
</aside>
```

while this:

```
#+ATTR_HTML: :controls controls :width 350
#+BEGIN_video
#+HTML: <source src="movie.mp4" type="video/mp4">
#+HTML: <source src="movie.ogg" type="video/ogg">
Your browser does not support the video tag.
#+END_video
```

exports to: 次へとエクスポートされんまう:

```
<video controls="controls" width="350">
  <source src="movie.mp4" type="video/mp4">
    <source src="movie.ogg" type="video/ogg">
      <p>Your browser does not support the video tag.</p>
</video>
```

When special blocks do not have ないときには a corresponding HTML5 element, the HTML exporter reverts to standard translation（~org-html-html5-elements~ を見てください）。For example, 例えば'`#+BEGIN_lederhosen`' は exports to `<div class="lederhosen">` へエクスポートします。

Special blocks cannot have headlines. For the HTML exporter to wrap the headline and its contents in `<section>` or `<article>` tags, set the '`HTML_CONTAINER`' property for the headline.

12.9.4 HTML preamble and postamble

The HTML exporter has delineations for preamble and postamble. The default value for `org-html-preamble` is `t`, which これは makes 作らせます the HTML exporter insert the preamble. See the variable `org-html-preamble-format` for the format string.

Set `org-html-preamble` to a string to override the default format string. If the string is a function, 関数であれば the HTML exporter expects the function この関数が to return 返すことを a string upon execution. The HTML exporter inserts 挿入します this string この文字列を in the preamble. The HTML exporter does not insert a preamble if `org-html-preamble` is set `nil`.

The default value デフォルト値は for `org-html-postamble` is `auto`, であり which これは makes the HTML exporter build a postamble from looking up author's name, email address, creator's name, and date. Set `org-html-postamble` to `t` to insert the postamble in the format specified in the `org-html-postamble-format` variable. The HTML exporter does not insert a postamble if `org-html-postamble` is set to `nil`.

12.9.5 Quoting HTML tags

The HTML エクスポートバックエンドは transforms 変換します'`<`' and '`>`' to '`<`' と'`>`' へ

To include raw HTML code in the Org file so the HTML エクスポートバックエンドは can insert 挿入することができます that HTML code in the output, use this inline syntax: このインラインの文法を'`@@html:...@@`'. 例です:

```
@@html:<b>@@bold text@@html:</b>@@
```

For larger raw HTML code blocks, use these HTML export code blocks:

```
#+HTML: Literal HTML code for export

#+BEGIN_EXPORT html
  All lines between these markers are exported literally
#+END_EXPORT
```

12.9.6 Headlines in HTML export

Headlines are exported to '`<h1>`', '`<h2>`', etc. Each headline gets the '`id`' attribute from '`CUSTOM_ID`' property, or a unique generated value, see 節 4.2 [Internal Links], ページ 40.

When `org-html-self-link-headlines` が is set to a non-`nil` value, `nil` でない値に the text of the headlines is also wrapped in '`<a>`' tags. These tags have a '`href`' attribute making the headlines link to themselves.

12.9.7 Links in HTML export

The HTML エクスポートバックエンドは transforms Org's internal links（節 4.2 [Internal Links], ページ 40, を見てください）to equivalent HTML links in the output. The back-end similarly handles Org's automatic links created 作成される by radio targets（節 4.3 [Radio Targets], ページ 41, を見てください）similarly. 同じように For Org links to external files, the back- end transforms 変換します the links to *relative* paths.

For Org links to other '`.org`' files, the back-end automatically 自動的に changes 変更します the file extension to '`.html`' へ and makes file paths relative. 相対的にします If the '`.org`' files have an equivalent '`.html`' version at the same location, then the converted links should work without any further manual intervention. However, しかし to disable 無効にするには this automatic path translation, set `org-html-link-org-files-as-html` to `nil`. When disabled, the HTML エクスポートバックエンドは substitutes th e ID-based links in the HTML output. For more about linking files when publishing to a directory, see 節 13.1.6 [Publishing links], ページ 206.

Org files can also have special directives to the HTML export back-end. For example, 例えば by using 使うことによって'`#+ATTR_HTML`' lines to specify new format attributes to `<a>` or `<img>` tags. This example shows 示しています changing the link's title and style:

```
#+ATTR_HTML: :title The Org-mode homepage :style color:red;
```

```
[[https://orgmode.org]]
```

12.9.8 Tables in HTML export

The HTML エクスポートバックエンドは uses 使いますorg-html-table-default-attributes を when exporting エクスポートするときにん Org tables Org-mode の表を to HTML へ By default, デフォルトで the exporter このエクスポーターは does not draw frames and cell borders. To change 変更するには for this for a table, use 使ってください the following lines before the table in the Org file:

```
#+CAPTION: This is a table with lines around and between cells
#+ATTR_HTML: border="2" rules="all" frame="border"
```

The HTML エクスポートバックエンドは preserves column groupings 列のグループを in Org tables（節 3.3 [Column Groups], ページ 25, を見てください）when exporting to HTML へエクスポートするときに

Additional options for customizing tables for HTML export.

`org-html-table-align-individual-fields`
: Non-`nil` `nil` でない値は attaches style attributes for alignment to each table field.

`org-html-table-caption-above`
: Non-`nil` `nil` でない値は places caption string at the beginning of the table.

`org-html-table-data-tags`
: Opening and ending tags for table data fields.

`org-html-table-default-attributes`
: Default attributes and values for table tags.

`org-html-table-header-tags`
: Opening and ending tags for table's header fields.

`org-html-table-row-tags`
: Opening and ending tags for table rows.

`org-html-table-use-header-tags-for-first-column`
: Non-`nil` `nil` でない値は formats 強制します column one in tables with header tags.

12.9.9 Images in HTML export

The HTML エクスポートバックエンドには has あります features to convert Org image links to HTML inline images and HTML clickable image links.

When the link in the Org file has no description, the HTML export back-end by default デフォルトで in-lines that image. 例です: '`[[file:myimg.jpg]]`' is in-lined, while 一方で'`[[file:myimg.jpg][the image]]`' は links リンクします to the text, '`the image`'. For more details, see the variable `org-html-inline-images`.

On the other hand, if the description part of the Org link is itself another link, such as '`file:`' or '`http:`' URL pointing to an image, the HTML エクスポートバックエンドは in-lines this image and links to the main image. This Org syntax enables the back-end to

link low-resolution thumbnail to the high-resolution version of the image, as shown in this example:

```
[[file:highres.jpg][file:thumb.jpg]]
```

To change 変更するには、attributes 属性を of in-lined images, インライン化されるイメージの use 使ってください'`#+ATTR_HTML`' 行を in the Org file. This example shows 示しています realignment to right, and adds `alt` and `title` attributes in support of text viewers and modern web accessibility standards.

```
#+CAPTION: A black cat stalking a spider
#+ATTR_HTML: :alt cat/spider image :title Action! :align right
[[./img/a.jpg]]
```

The HTML エクスポートバックエンドは copies コピーします the '`http`' links from the Org file as-is.

12.9.10 Math formatting in HTML export

LaTeX math snippets（節 11.5.1 [LaTeX fragments], ページ 139, を見てください）can be displayed 表示することができます in two different ways 2 つの異なる方法で on HTML pages. The default デフォルトあ is to use 使うということです the MathJax (`http://www.mathjax.org`) を which これは should work out of the box with Org[6][7]. Some MathJax display options can be configured 設定することができます via `org-html-mathjax-options`, or in the buffer. For example, 例えば with the following settings,

```
#+HTML_MATHJAX: align: left indent: 5em tagside: left font: Neo-Euler
#+HTML_MATHJAX: cancel.js noErrors.js
```

equation labels are displayed on the left margin and equations are five em from the left margin. In addition, 加えて it それっは loads the two MathJax extensions '`cancel.js`' and '`noErrors.js`'[8].

See 見てください the docstring ドキュメント文字列を of `org-html-mathjax-options` の for all supported variables. The MathJax のテンプレートを can be configure 設定することができます via `org-html-mathjax-template` をとおして

If you prefer, 好むのであれば you can also request 要求することもできます that LaTeX の断片が are processed 処理されることを into small images that will be inserted into the browser page. Before the availability of MathJax, this これが was the default method for Org files. This method requires that the dvipng program, dvisvgm or ImageMagick suite is available on your system. You can still get this processing with

```
#+OPTIONS: tex:dvipng
#+OPTIONS: tex:dvisvgm
```

or

```
#+OPTIONS: tex:imagemagick
```

---

[6] By default デフォルトで Org loads ロードします MathJax from cdnjs.com (`https://cdnjs.com`) as recommended by MathJax (`http://www.mathjax.org`).

[7] Please どうか note 注意してください that exported formulas are part of an HTML document, and that signs such as '`<`', '`>`', or '`&`' have special meanings. See MathJax TeX and LaTeX support (`http://docs.mathjax.org/en/latest/tex.html#tex-and-latex-in-html-documents`) を見てください。

[8] See TeX and LaTeX extensions (`http://docs.mathjax.org/en/latest/tex.html#tex-extensions`) in the MathJax manual (`http://docs.mathjax.org`) を見てください to learn 学ぶには about extensions. 拡張について

12.9.11 Text areas in HTML export

Before Org-mode's Babel, one popular approach to publishing code in HTML was by using 使うことによって‘`:textarea`’ を The advantage of this approach was that copying and pasting was built into browsers with simple JavaScript commands. Even editing before pasting was made simple.

The HTML エクスポートバックエンドは can create 作成することができます such text areas. It requires an ‘`#+ATTR_HTML`’ line as shown in the example below with the ‘`:textarea`’ option. This must be followed by either an example or a source code block. Other Org block types do not honor the ‘`:textarea`’ option.

By default, デフォルトで the HTML エクスポートバックエンドは creates 作成します a text area テキストエリアを 80 characters wide and height just enough to fit the content. Override these defaults with ‘`:width`’ and ‘`:height`’ options on the ‘`#+ATTR_HTML`’ line.

```
#+ATTR_HTML: :textarea t :width 40
#+BEGIN_EXAMPLE
  (defun org-xor (a b)
     "Exclusive or."
     (if a (not b) b))
#+END_EXAMPLE
```

12.9.12 CSS support

You can modify the CSS style definitions for the exported file. The HTML exporter assigns the following special CSS classes[9] to appropriate parts of the document---your style specifications may change these, in addition 加えて to any of the standard classes like for headlines, tables, etc.

`p.author`	author information, including email
`p.date`	publishing date
`p.creator`	creator info, about org mode version
`.title`	document title
`.subtitle`	document subtitle
`.todo`	TODO keywords, all not-done states
`.done`	the DONE keywords, all states that count DONE として
`.WAITING`	each TODO keyword also uses a class named after itself
`.timestamp`	timestamp
`.timestamp-kwd`	keyword associated with a timestamp, like ‘`SCHEDULED`’
`.timestamp-wrapper`	span around keyword plus timestamp
`.tag`	tag in a headline
`._HOME`	each tag uses itself as a class, "@" replaced by "_"
`.target`	target for links
`.linenr`	the line number in a code example
`.code-highlighted`	for highlighting referenced code lines
`div.outline-N`	div for outline level N (headline plus text)
`div.outline-text-N`	extra div for text at outline level N
`.section-number-N`	section number in headlines, different for each level

[9] If the classes on TODO keywords and tags lead to conflicts, use the variables 変数`org-html-todo-kwd-class-prefix` と`org-html-tag-class-prefix` to make them unique.

`.figure-number`	label like "Figure 1:"
`.table-number`	label like "Table 1:"
`.listing-number`	label like "Listing 1:"
`div.figure`	how to format an in-lined image
`pre.src`	formatted source code
`pre.example`	normal example
`p.verse`	verse paragraph
`div.footnotes`	footnote section headline
`p.footnote`	footnote definition paragraph, containing a footnote
`.footref`	a footnote reference number (always a <sup>)
`.footnum`	footnote number in footnote definition (always 常に<sup>)
`.org-svg`	default class for a linked '`.svg`' image

The HTML エクスポートバックエンドは includes 含めます a compact default style in each exported HTML file. To override the default style with another style, use these keywords in the Org file. They will replace the global defaults the HTML exporter uses. 使います

```
#+HTML_HEAD: <link rel="stylesheet" type="text/css" href="style1.css" />
#+HTML_HEAD_EXTRA: <link rel="alternate stylesheet" type="text/css" href="style2.css"
```

To just turn off the default style, customize カスタマイズしてくださいorg-html-head-include-default-style variable, or use this option line in the Org file.

```
#+OPTIONS: html-style:nil
```

For longer style definitions, either use several '`HTML_HEAD`' and '`HTML_HEAD_EXTRA`' keywords, or use `<style> ... </style>` blocks around them. Both of these approaches can avoid referring to an external file.

In order to add styles to a sub-tree, use the '`HTML_CONTAINER_CLASS`' property to assign a class to the tree. In order to specify CSS styles for a particular headline, you can use 使うことができます the specified 指定されている ID を in a '`CUSTOM_ID`' プロパティの中で You can also assign 割り当てることもできます a specific class 特定のクラスを to a headline 見出しに対して with the '`HTML_HEADLINE_CLASS`' property.

Never change the `org-html-style-default` constant. Instead かわりに use other simpler ways of customizing as described above.

12.9.13 JavaScript supported display of web pages

Sebastian Rose has written a JavaScript program especially designed to enhance the web viewing experience of HTML files created with Org. This program enhances large files in two different ways of viewing. One is an *Info*-like mode where each section is displayed separately and navigation can be done with the `n` and `p` keys, and some other keys as well, press `?` for an overview of the available keys. The second one has a *folding* view, much like Org provides 提供しています inside Emacs. Emacs の内側で The script is available at `https://orgmode.org/org-info.js` and the documentation at `https://orgmode.org/worg/code/org-info-js/`. The script is hosted on `https://orgmode.org`, but for reliability, prefer installing it on your own web server.

To use this program, just add this line to the Org file:

```
#+INFOJS_OPT: view:info toc:nil
```

The HTML header now has the code needed to automatically 自動的に invoke 実行するための the script. このスクリプトを For setting options, use the syntax from the above line for options described below:

‘`path:`’ The path to the script. The default is デフォルトは to grab the script from `https://orgmode.org/org-info.js`, but you might want to have a local copy and use a path like ‘`../scripts/org-info.js`’.

‘`view:`’ Initial view when the website is first shown. Possible values are:

‘`info`’	Info-like interface with one section per page
‘`overview`’	Folding interface, initially showing only top-level
‘`content`’	Folding interface, starting with all headlines visible
‘`showall`’	Folding interface, all headlines and text visible

‘`sdepth:`’ Maximum headline level still considered as an independent section for info and folding modes. The default デフォルトは is taken 取られます from `org-export-headline-levels`, i.e., すなわち the ‘`H`’ switch in ‘`OPTIONS`’. If this これが is smaller than in `org-export-headline-levels`, each info/folding section can still contain child headlines.

‘`toc:`’ Should the table of contents *initially* be visible? Even when ‘`nil`’, you can always 常に get to the "toc" with `i`.

‘`tdepth:`’ The depth of the table of contents. The defaults デフォルトは are taken 取られます from the variables `org-export-headline-levels` and `org-export-with-toc`.

‘`ftoc:`’ Does the CSS of the page specify a fixed position for the "toc"? If yes, イエスであれば the toc is displayed as a section.

‘`ltoc:`’ Should there be short contents (children) in each section? Make this ‘`above`’ if the section should be above initial text.

‘`mouse:`’ Headings are highlighted when the mouse is over them. Should be ‘`underline`’ (default) or a background color like ‘`#cccccc`’.

‘`buttons:`’
Should view-toggle buttons be everywhere? When ‘`nil`’ (the default), only one such button is present.

You can choose default values for these options by customizing カスタマイズすることによって the variable 変数`org-infojs-options` を If you always 常に want to apply 適用したいのであれば the script このスクリプトを to your pages, configure 設定してください the 変数`org-export-html-use-infojs` を

12.10 LaTeX Export

The LaTeX エクスポートバックエンド can handle complex documents, incorporate standard or custom LaTeX document classes, generate documents using alternate LaTeX engines, and produce fully linked PDF files with indexes, bibliographies, and tables of contents, destined for interactive online viewing or high-quality print publication.

While the details are covered in-depth in this section, here are some quick references to variables for the impatient: for engines, see `org-latex-compiler`; for build sequences,

see `org-latex-pdf-process`; for packages, see `org-latex-default-packages-alist` and `org-latex-packages-alist`.

An important note about the LaTeX エクスポートバックエンド: it is sensitive to blank lines in the Org document. That's because LaTeX itself depends on blank lines to tell apart syntactical elements, such as paragraphs.

12.10.1 LaTeX/PDF export commands

C-c C-e l l (`org-latex-export-to-latex`)
: Export エクスポートします to a LaTeX ファイルへ with a '`.tex`' という拡張子をもつ For '`myfile.org`' に対して、Org Org-mode は exports to '`myfile.tex`' へエクスポートし、overwriting without warning. 警告なしで上書きします。

C-c C-e l L (`org-latex-export-as-latex`)
: Export エクスポートします。to a temporary buffer. 一時的なバッファへ Do not create a file. ファイルを作成しません。

C-c C-e l p (`org-latex-export-to-pdf`)
: Export エクスポートします as LaTeX ファイルとして and convert it それを to PDF file.

C-c C-e l o
: Export エクスポートします as LaTeX として and convert it それを to PDF へ then open 開きます the PDF この PDF を using the default viewer. デフォルトのビューワーを使って

M-x org-export-region-as-latex
: Convert the region リージョンを to LaTeX under the assumption that it was in Org mode syntax before. This is a global command that can be invoked in any buffer.

The LaTeX エクスポートバックエンド can use 使うことができます any いずれかを of these LaTeX engines: '`pdflatex`' と'`xelatex`' と'`lualatex`' です。These engines compile LaTeX files with different compilers, packages, and output options. The LaTeX エクスポートバックエンド finds the compiler version to use from `org-latex-compiler` variable or the '`#+LATEX_COMPILER`' keyword in the Org file. See 見てください the docstring for the `org-latex-default-packages-alist` for loading packages with certain compilers. Also see `org-latex-bibtex-compiler` も見てください。to set 設定するには the bibliography compiler[fn:131].

12.10.2 LaTeX specific export settings

The LaTeX エクスポートバックエンド has several additional keywords for customizing LaTeX output. Setting these keywords これらのキーワードの設定は works similar 同じように動きます to the general options 一般的な設定と（節 12.2 [Export Settings], ページ 146, を見てください）。

'`DESCRIPTION`'
: The document's description. The description 説明が along with ともに author name, keywords, and related file metadata are inserted in the output file by the hyperref package. See 見てください`org-latex-hyperref-template` for

customizing metadata items. See 見てくださいorg-latex-title-command for typesetting description into the document's front matter. Use 使ってください multiple 'DESCRIPTION' keywords for long descriptions.

'LANGUAGE'
: In order to be effective, the 'babel' or 'polyglossia' packages---according 従って to the LaTeX compiler used---must be loaded with the appropriate language as argument. This can be accomplished by modifying the 'org-latex-package-alist' variable, e.g., 例えば with the following snippet:

```
(add-to-list org-latex-package-alist
             '("AUTO" "babel" t ("pdflatex")))
(add-to-list org-latex-package-alist
             '("AUTO" "polyglossia" t ("xelatex" "lualatex")))
```

'LATEX_CLASS'
: This is LaTeX document class, such as *article*, *report*, *book*, and so on, which これは contain 含んでいます predefined preamble and headline level mapping that the LaTeX エクスポートバックエンドが必用とする The back-end このバックエンドは reads the default class name from the org-latex-default-class variable. Org has *article* as the default class. A valid default class must be an element of org-latex-classes.

'LATEX_CLASS_OPTIONS'
: Options the LaTeX エクスポートバックエンド uses 使います when calling the LaTeX document class.

'LATEX_COMPILER'
: The compiler, such as 'pdflatex', 'xelatex', 'lualatex', for producing the PDF. See org-latex-compiler.

'LATEX_HEADER'
'LATEX_HEADER_EXTRA'
: Arbitrary lines to add to the document's preamble, before the hyperref settings. See org-latex-classes for adjusting the structure and order of the LaTeX headers.

'KEYWORDS'
: The keywords for the document. The description along with ともに author name, 著者名、keywords, and related file metadata are inserted in the output file by the hyperref package. See org-latex-hyperref-template for customizing metadata items. See org-latex-title-command for typesetting description into the document's front matter. Use 使ってください multiple 'KEYWORDS' lines if necessary. 必用であれば

'SUBTITLE'
: The document's subtitle. It is typeset as per org-latex-subtitle-format. If org-latex-subtitle-separate がnil でない値の場合には、it is typed as part of the \title macro. See org-latex-hyperref-template for customizing metadata items. See org-latex-title-command for typesetting description into the document's front matter.

The following sections have further details.

12.10.3 LaTeX header and sectioning structure

The LaTeX エクスポートバックエンドは converts 変換します the first three of Org's outline levels into LaTeX headlines. The remaining Org levels 残りの Org-mode の見出しのレベルは are exported エクスポートされます as lists. リストとして To change 変更するには this これを globally グローバルに for the cut-off point このカットオフポイントを between levels and lists, 見出しのレベルとリストの間の（節 12.2 [Export Settings], ページ 146, を見てください）。

By default, デフォルトで the LaTeX エクスポートバックエンド uses 使います。the *article* ドキュメントクラスを

To change 変更するには the default class globally, グローバルに edit `org-latex-default-class` を編集してください。To change 変更するには the default class locally ローカルに in an Org file, add option lines '`#+LATEX_CLASS: myclass`'. To change 変更するには the default class for just a part of the Org file, set a sub-tree property, '`EXPORT_LATEX_CLASS`'. The class name entered here must be valid member of `org-latex-classes`. This variable defines a header template for each class into which the exporter splices the values of `org-latex-default-packages-alist` and `org-latex-packages-alist`. Use 使ってください the same three variables to define custom sectioning or custom classes.

The LaTeX エクスポートバックエンド sends the '`LATEX_CLASS_OPTIONS`' keyword and '`EXPORT_LATEX_CLASS_OPTIONS`' property as options to the LaTeX `\documentclass` macro. The options and the syntax for specifying them, including 含む enclosing them in square brackets, follow LaTeX conventions.

```
#+LATEX_CLASS_OPTIONS: [a4paper,11pt,twoside,twocolumn]
```

The LaTeX エクスポートバックエンドは appends 追加します values from キーワード'`LATEX_HEADER`' と'`LATEX_HEADER_EXTRA`' からの to the LaTeX header. The docstring ドキュメント文字列が for `org-latex-classes` explains in more detail. より詳細に説明しています Also また note 注意してください that LaTeX エクスポートバックエンドは does not append 追加しません'`LATEX_HEADER_EXTRA`' to the header when previewing LaTeX snippets（節 11.5.2 [Previewing LaTeX fragments], ページ 140, を見てください）。

A sample Org file with the above headers:

```
#+LATEX_CLASS: article
#+LATEX_CLASS_OPTIONS: [a4paper]
#+LATEX_HEADER: \usepackage{xyz}

* Headline 1
  some text
* Headline 2
  some more text
```

12.10.4 Quoting LaTeX code

The LaTeX エクスポートバックエンド can insert 挿入することができます any arbitrary LaTeX code, 任意のLaTeX のコードを see 節 11.5 [Embedded LaTeX], ページ 139, を見てください There are three ways to embed such code in the Org file and they all use different quoting syntax.

Inserting in-line quoted with @ symbols:

```
Code embedded in-line @@latex:any arbitrary LaTeX code@@ in a paragraph.
```

Inserting as one or more keyword lines in the Org file:

```
#+LATEX: any arbitrary LaTeX code
```

Inserting as an export block in the Org file, where そこでは the back-end exports エクスポートします any code between begin and end markers:

```
#+BEGIN_EXPORT latex
  any arbitrary LaTeX code
#+END_EXPORT
```

12.10.5 Tables in LaTeX export

The LaTeX エクスポートバックエンド can pass several LaTeX attributes for table contents and layout. Besides specifying a label（節 4.2 [Internal Links], ページ 40, を見てください）and a caption（節 11.8 [Captions], ページ 144, を見てください）the other valid LaTeX attributes include:

‘`:mode`’ The LaTeX エクスポートバックエンド wraps the table differently depending 依存して on the mode モードに for accurate rendering of math symbols. Mode is either ‘`table`’, ‘`math`’, ‘`inline-math`’ or ‘`verbatim`’.

For ‘`math`’ or ‘`inline-math`’ mode, LaTeX エクスポートバックエンド wraps the table in a math environment, but every cell in it is exported as-is. The LaTeX エクスポートバックエンド determines the default mode from `org-latex-default-table-mode`. The LaTeX エクスポートバックエンド merges contiguous tables in the same mode into a single environment.

‘`:environment`’
Set 設定します the default LaTeX table environment for the LaTeX エクスポートバックエンドが to use 使う when exporting エクスポートするときに Org tables. Org-mode の表を Common LaTeX table environments are provided by these packages: tabularx, longtable, array, tabu, and bmatrix. For packages, such as tabularx and tabu, or any newer replacements, include them in the `org-latex-packages-alist` variable so the LaTeX エクスポートバックエンド can insert 挿入することができます the appropriate load package headers in the converted LaTeX file. Look in the docstring for the `org-latex-packages-alist` variable for configuring these packages for LaTeX snippet previews, if any.

‘`:caption`’
Use 使います‘`CAPTION`’ keyword to set a simple caption for a table（節 11.8 [Captions], ページ 144, を見てください）。For custom captions, use ‘`:caption`’ attribute, which これは accepts 受け入れます raw LaTeX code. ‘`:caption`’ value overrides ‘`CAPTION`’ value.

‘`:float`’
‘`:placement`’
The table environments by default デフォルトで are not floats in LaTeX. To make them floating objects use ‘`:float`’ with one of the following options: ‘`sideways`’, ‘`multicolumn`’, ‘`t`’, and ‘`nil`’.

LaTeX floats can also have additional layout ‘`:placement`’ attributes. These are the usual ‘`[h t b p ! H]`’ permissions specified in square brackets. Note 注意し

てください that for ‘:float sideways’ tables, the LaTeX エクスポートバックエンド ignores ‘:placement’ attributes.

‘:align’
‘:font’
‘:width’ The LaTeX エクスポートバックエンド uses these attributes for regular tables to set their alignments, fonts, and widths.

‘:spread’ When ‘:spread’ is non-nil, nil でない値のときには、the LaTeX エクスポートバックエンド spreads or shrinks the table by the ‘:width’ for tabu and longtabu environments. ‘:spread’ has no effect if ‘:width’ is not set.

‘:booktabs’
‘:center’
‘:rmlines’
All three commands are toggles. ‘:booktabs’ brings in modern typesetting enhancements to regular tables. The booktabs package has to be loaded through org-latex-packages-alist. ‘:center’ is for centering the table. ‘:rmlines’ removes all but the very first horizontal line made of ASCII characters from "table.el" tables only.

‘:math-prefix’
‘:math-suffix’
‘:math-arguments’
The LaTeX エクスポートバックエンドは inserts 挿入します‘:math-prefix’ string value in a math environment before the table. The LaTeX エクスポートバックエンド inserts 挿入します‘:math-suffix’ string value in a math environment after the table. The LaTeX エクスポートバックエンドは inserts 挿入します‘:math-arguments’ string value between the macro name and the table's contents. ‘:math-arguments’ comes in use for matrix macros that require more than one argument, such as ‘qbordermatrix’.

LaTeX table attributes help formatting tables for a wide range of situations, such as matrix product or spanning multiple pages:

```
#+ATTR_LATEX: :environment longtable :align l|lp{3cm}r|l
| ... | ... |
| ... | ... |

#+ATTR_LATEX: :mode math :environment bmatrix :math-suffix \times
| a | b |
| c | d |
#+ATTR_LATEX: :mode math :environment bmatrix
| 1 | 2 |
| 3 | 4 |
```

Set the caption with the LaTeX command ‘\bicaption{HeadingA}{HeadingB}’:

```
#+ATTR_LATEX: :caption \bicaption{HeadingA}{HeadingB}
| ... | ... |
| ... | ... |
```

12.10.6 Images in LaTeX export

The LaTeX エクスポートバックエンド processes image links in Org files that do not have descriptions, such as these links '`[[file:img.jpg]]`' or '`[[./img.jpg]]`', as direct image insertions in the final PDF output. In the PDF, they are no longer links but actual images embedded on the page. The LaTeX エクスポートバックエンド uses '`\includegraphics`' macro to insert the image. But しかし for TikZ (`http://sourceforge.net/projects/pgf/`) images, the back-end uses 使います an `\input` macro wrapped within a `tikzpicture` environment.

For specifying image '`:width`', '`:height`', and other '`:options`', use this syntax:

```
#+ATTR_LATEX: :width 5cm :options angle=90
[[./img/sed-hr4049.pdf]]
```

For custom commands for captions, use the '`:caption`' attribute. It overrides the default '`#+CAPTION`' value:

```
#+ATTR_LATEX: :caption \bicaption{HeadingA}{HeadingB}
[[./img/sed-hr4049.pdf]]
```

When captions follow the method as described in 節 11.8 [Captions], ページ 144, the LaTeX エクスポートバックエンドは wraps ラップします the picture in a floating '`figure`' environment. To float an image without specifying a caption, set the '`:float`' attribute to one of the following:

'`t`'
: For a standard '`figure`' environment; used by default デフォルトで whenever いつでも an image has a caption.

'`multicolumn`'
: To span the image across multiple columns of a page; the back-end wraps the image in a '`figure*`' environment.

'`wrap`'
: For text to flow around the image on the right; the figure occupies the left half of the page.

'`sideways`'
: For a new page with the image sideways, rotated ninety degrees, in a '`sidewaysfigure`' environment; overrides '`:placement`' setting.

'`nil`'
: To avoid a '`:float`' even if using a caption.

Use 使ってください the '`placement`' attribute to modify a floating environment's placement.

```
#+ATTR_LATEX: :float wrap :width 0.38\textwidth :placement {r}{0.4\textwidth}
[[./img/hst.png]]
```

The LaTeX export back-nd centers all images by default. デフォルトで Setting '`:center`' to '`nil`' disables centering. To disable centering globally, set `org-latex-images-centered` to '`t`'.

Set the '`:comment-include`' attribute to non-`nil` value `nil` でない値に for the LaTeX エクスポートバックエンドが to comment out the '`\includegraphics`' macro.

12.10.7 Plain lists in LaTeX export

The LaTeX エクスポートバックエンド accepts the '`environment`' and '`options`' attributes for plain lists. Both attributes work together for customizing lists, as shown in the examples:

```
#+LATEX_HEADER: \usepackage[inline]{enumitem}
Some ways to say "Hello":
#+ATTR_LATEX: :environment itemize*
#+ATTR_LATEX: :options [label={}, itemjoin={,}, itemjoin*={, and}]
- Hola
- Bonjour
- Guten Tag.
```

Since LaTeX が supports サポートしています only four levels of nesting for lists, use an external package, such as '`enumitem`' in LaTeX, for levels deeper than four:

```
#+LATEX_HEADER: \usepackage{enumitem}
#+LATEX_HEADER: \renewlist{itemize}{itemize}{9}
#+LATEX_HEADER: \setlist[itemize]{label=$\circ$}
- One
  - Two
    - Three
      - Four
        - Five
```

12.10.8 Source blocks in LaTeX export

The LaTeX エクスポートバックエンド can make source code blocks into floating objects through the attributes '`:float`' and '`:options`'. For '`:float`':

'`t`' Makes a source block float; by default デフォルトで floats any source block with a caption.

'`multicolumn`'
Spans the source block across multiple columns of a page.

'`nil`' Avoids a '`:float`' even if using a caption; useful 役に立ちます for source code blocks that may not fit on a page.

```
#+ATTR_LATEX: :float nil
#+BEGIN_SRC emacs-lisp
  Lisp code that may not fit in a single page.
#+END_SRC
```

The LaTeX エクスポートバックエンド passes string values in '`:options`' to LaTeX packages for customization of that specific source block. In the example below, the '`:options`' are set for Minted. Minted is a source code highlighting LaTeX package with many configurable options.

```
#+ATTR_LATEX: :options commentstyle=\bfseries
#+BEGIN_SRC emacs-lisp
  (defun Fib (n)
    (if (< n 2) n (+ (Fib (- n 1)) (Fib (- n 2)))))
#+END_SRC
```

To apply similar configuration options for all source blocks in a file, use the `org-latex-listings-options` and `org-latex-minted-options` variables.

12.10.9 Example blocks in LaTeX export

The LaTeX エクスポートバックエンド wraps the contents of example blocks in a '`verbatim`' environment. To change 変更するには this behavior to use another environment globally, グローバルに specify an appropriate export filter（節 12.17 [Advanced Export Configuration], ページ 196, を見てください）。To change 変更するには this behavior このふるまいを to use 使うように another environment 別の環境を for each block, use 使ってください the '`:environment`' パラメータを to specify 指定するために a custom environment. カスタムの環境を

```
#+ATTR_LATEX: :environment myverbatim
#+BEGIN_EXAMPLE
  This sentence is false.
#+END_EXAMPLE
```

12.10.10 Special blocks in LaTeX export

For other special blocks in the Org file, the LaTeX エクスポートバックエンド makes a special environment of the same name. The back-end also takes '`:options`', if any, and appends as-is to that environment's opening string. 例です:

```
#+BEGIN_abstract
  We demonstrate how to solve the Syracuse problem.
#+END_abstract

#+ATTR_LATEX: :options [Proof of important theorem]
#+BEGIN_proof
  ...
  Therefore, any even number greater than 2 is the sum of two primes.
#+END_proof
```

exports to 次へとエクスポートします

```
\begin{abstract}
  We demonstrate how to solve the Syracuse problem.
\end{abstract}

\begin{proof}[Proof of important theorem]
  ...
  Therefore, any even number greater than 2 is the sum of two primes.
\end{proof}
```

If you need to insert 挿入する必用があれば a specific caption command, 特定のキャプションコマンドを use 使ってください。'`:caption`' 属性を It それは overrides 上書きします standard '`CAPTION`' value, if any. For example: 例です:

```
#+ATTR_LATEX: :caption \MyCaption{HeadingA}
#+BEGIN_proof
  ...
#+END_proof
```

12.10.11 Horizontal rules in LaTeX export

The LaTeX エクスポートバックエンドは converts 変換します horizontal rules 垂直線を by the specified '`:width`' and '`:thickness`' attributes. 例です:

```
#+ATTR_LATEX: :width .6\textwidth :thickness 0.8pt
-----
```

12.11 Markdown Export

The Markdown エクスポートバックエンド"md" は converts 変換します an Org file to Markdown format, as defined at `http://daringfireball.net/projects/markdown/`.

Since it is built on top of the HTML back-end（節 12.9 [HTML Export], ページ 159, を見てください）、it それは converts 変換します every Org construct not defined in Markdown syntax, such as tables, to HTML.

Markdown export commands

C-c C-e m m (`org-md-export-to-markdown`)
: Export エクスポートします to a text file テキストファイルへ with Markdown の文法を持つ For '`myfile.org`' に対して Org Org mode は exports to '`myfile.md`' へエクスポートします overwritten without warning.

C-c C-e m M (`org-md-export-as-markdown`)
: Export エクスポートします to a temporary buffer. 一時的なファイルへ Does not create a file. ファイルを作成しません。

C-c C-e m o
: Export エクスポートします as a text file テキストファイルとして with Markdown syntax, then open it. それを開きます。

Header and sectioning structure

Based on `org-md-headline-style`, Markdown export can generate headlines of both *atx* and *setext* types. *atx* limits headline levels to two whereas *setext* limits headline levels to six. Beyond these limits, このリミットを超えると the export back-end このエクスポートバックエンドは converts 変換します headlines 見出しを to lists. リストへ To set 設定するには a limit to a level before the absolute limit（節 12.2 [Export Settings], ページ 146, を見てください）。

12.12 OpenDocument Text Export

The ODT エクスポートバックエンドは handles 扱います creating of OpenDocument Text (ODT) format. Documents created by this exporter use the *OpenDocument-v1.2 specification*[10] and are compatible with LibreOffice 3.4.

12.12.1 Pre-requisites for ODT export

The ODT エクスポートバックエンドは relies on the zip program to create the final compressed ODT output. Check if '`zip`' is locally available and executable. Without it, export cannot finish.

[10] Open Document Format for Office Applications (OpenDocument) Version 1.2 (`http://docs.oasis-open.org/office/v1.2/OpenDocument-v1.2.html`) を見てください。

12.12.2 ODT export commands

C-c C-e o o (org-export-to-odt)
: Export エクスポートします。as OpenDocument Text ファイルとして

  If org-odt-preferred-output-format が is specified, 指定される場合には the ODT エクスポートバックエンドは automatically 自動的に converts 変換します the exported file エクスポートされるファイルを to that format.

  For 'myfile.org' に対して、Org Org-mode は exports to 'myfile.odt' へエクスポートし、overwriting without warning. 警告なしで上書きします。The ODT エクスポートバックエンドは exports エクスポートします a region リージョンを only if a region was active.

  If the selected region is a single tree, the ODT エクスポートバックエンドは makes the tree head the document title. Incidentally, *C-c @* selects the current sub-tree. If the tree head entry has, or inherits, an 'EXPORT_FILE_NAME' property, the ODT エクスポートバックエンドは uses 使います that for file name.

C-c C-e o O
: Export エクスポートし、as an OpenDocument Text ファイルとして and open the resulting file. 結果のファイルを開きます。

  If org-export-odt-preferred-output-format が is specified, 指定されている場合には open 開きます the converted file instead. かわりに[Automatically exporting to other formats], ページ 178, を見てください。

12.12.3 ODT specific export settings

The ODT エクスポートバックエンドには has あります several additional keywords for customizing ODT output. Setting these keywords これらのキーワードの設定は works similar 同じように動きます to the general options 一般的なオプションと（節 12.2 [Export Settings], ページ 146, を見てください）。

'DESCRIPTION'
: This これは is the document's description, which これを the ODT エクスポートバックエンドは inserts 挿入します as document metadata. For long descriptions, use multiple lines, prefixed with 'DESCRIPTION'.

'KEYWORDS'
: The keywords for the document. The ODT エクスポートバックエンドは inserts 挿入します the description この説明を along with ともに author name, 著者名、keywords, and related file metadata as metadata in the output file. Use 使ってください multiple 'KEYWORDS' if necessary. 必用であれば

'ODT_STYLES_FILE'
: The ODT エクスポートバックエンドは uses 使います the org-odt-styles-file by default. デフォルトで See 節 12.12.5 [Applying custom styles], ページ 178, を見てください。for details. 詳細については

'SUBTITLE'
: The document subtitle.

12.12.4 Extending ODT export

The ODT エクスポートバックエンドは can produce 生成することができます documents in other formats besides ODT using a specialized ODT converter process. Its common interface works with popular converters to produce formats such as '`doc`', or convert a document from one format, say '`csv`', to another format, say '`xls`'.

Customize カスタマイズしてくださいorg-odt-convert-process variable to point to '`unoconv`', which is the ODT's preferred converter. Working installations of LibreOffice would already have '`unoconv`' installed. Alternatively, other converters may be substituted here. See [Configuring a document converter], ページ 183.

Automatically exporting to other formats

If ODT format is just an intermediate step to get to other formats, such as '`doc`', '`docx`', '`rtf`', or '`pdf`', etc., then extend the ODT エクスポートバックエンドを to directly produce that format. Specify the final format in the `org-odt-preferred-output-format` variable. This is one way to extend（節 12.12.2 [ODT export commands], ページ 177, を見てください）。

Converting between document formats

The Org export back-end is made to be inter-operable with a wide range of text document format converters. Newer generation converters, such as LibreOffice and Pandoc, can handle hundreds of formats at once. Org provides 提供しています a consistent interaction with whatever converter is installed. Here are some generic commands: これらはいくつかの一般的なコマンドです:

M-x org-odt-convert
: Convert an existing document from one format to another. With a prefix argument, 1 つの前置引数といっしょだしたお opens 開きます the newly produced file.

12.12.5 Applying custom styles

The ODT export back-end comes with many OpenDocument styles（[Working with OpenDocument style files], ページ 183, を見てください）。To expand or further customize these built-in style sheets, either edit the style sheets directly or generate them using an application such as LibreOffice. The example here shows 示しています creating a style using LibreOffice.

Applying custom styles: the easy way

1. Create 作成します a sample '`example.org`' file with settings as shown below, and export it to ODT format.

   ```
   #+OPTIONS: H:10 num:t
   ```

2. Open the above '`example.odt`' using LibreOffice. Use 使ってください the *Stylist* to locate the target styles, which これは typically 通常は have the "Org" prefix. Open one, modify, and save as either OpenDocument Text (ODT) or OpenDocument Template (OTT) file.

3. Customize カスタマイズしまづ the variable 変数`org-odt-styles-file` を and point it to the newly created file. For additional configuration options, see [Overriding factory styles], ページ 184, を見てください

   To apply an ODT style to a particular file, use the '`ODT_STYLES_FILE`' keyword as shown in the example below:

   ```
   #+ODT_STYLES_FILE: "/path/to/example.ott"
   ```

   or

   ```
   #+ODT_STYLES_FILE: ("/path/to/file.ott" ("styles.xml" "image/hdr.png"))
   ```

Using third-party styles and templates

The ODT export back-end relies on many templates and style names. Using third-party styles and templates can lead to mismatches. Templates derived from built in ODT templates and styles seem to have fewer problems.

12.12.6 Links in ODT export

ODT exporter creates 作成します native cross-references for internal links. It creates 作成します Internet-style links for all other links.

A link with no description and pointing to a regular, un-itemized, outline heading is replaced with a cross-reference and section number of the heading.

A '`\ref{label}`'-style reference to an image, table etc., is replaced with a cross-reference and sequence number of the labeled entity. See 節 12.12.10 [Labels and captions in ODT export], ページ 182.

12.12.7 Tables in ODT export

The ODT export back-end handles native Org-mode tables（章 3 [Tables], ページ 19, を見てください）and simple '`table.el`' tables. Complex '`table.el`' tables having column or row spans are not supported. Such tables are stripped from the exported document.

By default, デフォルトで the ODT export back-end exports エクスポートします a table with top and bottom frames and with ruled lines separating row and column groups（節 3.3 [Column Groups], ページ 25, を見てください）。All tables are typeset to occupy the same width. The ODT export back-end honors any table alignments and relative widths for columns（節 3.2 [Column Width and Alignment], ページ 23, を見てください）。

Note 注意してください that the ODT エクスポートバックエンドが interprets column widths as weighted ratios, the default weight being 1.

Specifying '`:rel-width`' property on an '`ATTR_ODT`' line controls the width of the table. 例です:

```
#+ATTR_ODT: :rel-width 50
| Area/Month    |   Jan |   Feb |   Mar |   Sum |
|---------------+-------+-------+-------+-------|
| /             |     < |       |       |     < |
| <l13>         |  <r5> |  <r5> |  <r5> |  <r6> |
| North America |     1 |    21 |   926 |   948 |
| Middle East   |     6 |    75 |   844 |   925 |
| Asia Pacific  |     9 |    27 |   790 |   826 |
```

```
|---------------+-------+-------+-------+-------|
| Sum           |    16 |   123 |  2560 |  2699 |
```

On export, the above table takes 50% of text width area. The exporter sizes the columns in the ratio: 13:5:5:5:6. The first column is left-aligned and rest of the columns, right-aligned. Vertical rules separate the header and the last column. Horizontal rules separate the header and the last row.

For even more customization, create custom table styles and associate them with a table using the '`ATTR_ODT`' keyword. See [Customizing tables in ODT export], ページ 185.

12.12.8 Images in ODT export

Embedding images

The ODT エクスポートバックエンドは processes 処理します image links イメージへのリンクを in Org files that do not have descriptions, such as these links '`[[file:img.jpg]]`' or '`[[./img.jpg]]`', as direct image insertions in the final output. Either of these examples works:

```
[[file:img.png]]
[[./img.png]]
```

Embedding clickable images

For clickable images, provide a link whose description is another link to an image file. For example, 例えば to embed an image '`org-mode-unicorn.png`' which when clicked jumps to `https://orgmode.org` website, do the following

```
[[https://orgmode.org][./org-mode-unicorn.png]]
```

Sizing and scaling of embedded images

Control the size and scale of the embedded images with the '`ATTR_ODT`' attribute.

The ODT export エクスポートバックエンドは starts はじめます with establishing the size of the image in the final document. The dimensions of this size are measured in centimeters. The back-end then queries the image file for its dimensions measured in pixels. For this measurement, the back-end relies on ImageMagick's identify program or Emacs `create-image` and `image-size` API. ImageMagick is the preferred choice for large file sizes or frequent batch operations. The back-end このバックエンドは then converts 変換します the pixel dimensions using `org-odt-pixels-per-inch` into the familiar 72 dpi or 96 dpi. The default value デフォルト値は for this これに対する is in `display-pixels-per-inch` です which これを can be tweaked 変更することができます for better results based on the capabilities of the output device. Here これらは are some common image scaling operations:

Explicitly size the image

To embed '`img.png`' as a 10 cm x 10 cm image, do the following:

```
#+ATTR_ODT: :width 10 :height 10
[[./img.png]]
```

Scale the image

To embed '`img.png`' at half its size, do the following:

```
#+ATTR_ODT: :scale 0.5
```

```
[[./img.png]]
```

Scale the image to a specific width

To embed '`img.png`' with a width of 10 cm while retaining the original height:width ratio, do the following:

```
#+ATTR_ODT: :width 10
[[./img.png]]
```

Scale the image to a specific height

To embed '`img.png`' with a height of 10 cm while retaining the original height:width ratio, do the following:

```
#+ATTR_ODT: :height 10
[[./img.png]]
```

Anchoring of images

The ODT export back-end can anchor images to '`as-char`', '`paragraph`', or '`page`'. Set the preferred anchor using the '`:anchor`' property of the '`ATTR_ODT`' line.

To create 作成するには an image イメージを that is anchored to a page:

```
#+ATTR_ODT: :anchor page
[[./img.png]]
```

12.12.9 Math formatting in ODT export

The ODT exporter has special support for handling math.

12.12.9.1 LaTeX math snippets

LaTeX math snippets（節 11.5.1 [LaTeX fragments], ページ 139, を見てください）can be embedded in the ODT document in one of the following ways:

MathML Add this line to the Org file. This option is activated on a per-file basis. ファイルごとに

```
#+OPTIONS: tex:t
```

With this option, LaTeX fragments are first converted into MathML fragments using an external LaTeX-to-MathML converter program. The resulting MathML fragments are then embedded as an OpenDocument Formula in the exported document.

You can specify 指定することができます the LaTeX-to-MathML converter by customizing カスタマイズすることによって the 変数`org-latex-to-mathml-convert-command` と`org-latex-to-mathml-jar-file` を

If you prefer to use MathToWeb[11] as your converter, you can configure 設定することあできます the above variables as shown below.

```
(setq org-latex-to-mathml-convert-command
      "java -jar %j -unicode -force -df %o %I"
      org-latex-to-mathml-jar-file
      "/path/to/mathtoweb.jar")
```

or, to use LaTeXML[12] instead, かわりに

[11] See MathToWeb (`http://www.mathtoweb.com/cgi-bin/mathtoweb_home.pl`) を見てください。

[12] See `http://dlmf.nist.gov/LaTeXML/` を見てください。

```
(setq org-latex-to-mathml-convert-command
      "latexmlmath \"%i\" --presentationmathml=%o")
```

To quickly 素早く verify the reliability of the LaTeX-to-MathML converter, use the following commands:

M-x org-export-as-odf
: Convert a LaTeX math snippet to an OpenDocument formula (‘`.odf`’) file.

M-x org-export-as-odf-and-open
: Convert a LaTeX math snippet to an OpenDocument formula (‘`.odf`’) file and open the formula file with the system-registered application.

PNG images
: Add this line to the Org file. This option is activated on a per-file basis. ファイルごとに

```
#+OPTIONS: tex:dvipng
```

```
#+OPTIONS: tex:dvisvgm
```

or

```
#+OPTIONS: tex:imagemagick
```

Under this option, LaTeX fragments are processed into PNG or SVG images and the resulting images are embedded in the exported document. This method requires dvipng program, dvisvgm or ImageMagick programs.

12.12.9.2 MathML and OpenDocument formula files

When embedding LaTeX math snippets in ODT documents is not reliable, there is one more option to try. Embed an equation by linking to its MathML (‘`.mml`’) source or its OpenDocument formula (‘`.odf`’) file as shown below:

```
[[./equation.mml]]
```

or

```
[[./equation.odf]]
```

12.12.10 Labels and captions in ODT export

ODT format handles labeling and captioning of objects based on their types. Inline images, tables, LaTeX fragments, and Math formulas are numbered and captioned separately. Each object also gets a unique sequence number based on its order of first appearance in the Org file. Each category has its own sequence. A caption is just a label applied to these objects.

```
#+CAPTION: Bell curve
#+NAME:   fig:SED-HR4049
[[./img/a.png]]
```

When rendered, it may show as follows in the exported document:

```
Figure 2: Bell curve
```

To modify the category component of the caption, customize カスタマイズしてください the option オプション`org-odt-category-map-alist` を For example, 例えば to

tag embedded images with the string "Illustration" instead かわりに of the default string "Figure", use the following setting: 次の設定を使ってください:

```
(setq org-odt-category-map-alist
      '(("__Figure__" "Illustration" "value" "Figure" org-odt--enumerable-image-p)))
```

With the above modification, the previous example changes to:

```
Illustration 2: Bell curve
```

12.12.11 Literal examples in ODT export

The ODT export back-end supports サポートしています literal examples（節 11.6 [Literal Examples], ページ 141, を見てください）with full fontification. Internally, the ODT export back-end relies on '`htmlfontify.el`' to generate the style definitions needed for fancy listings. The auto-generated styles get '`OrgSrc`' prefix and inherit colors from the faces used by Emacs Font Lock library for that source language.

For custom fontification styles, customize カスタマイズしてください the `org-odt-create-custom-styles-for-srcblocks` オプションを

To turn off fontification of literal examples, customize カスタマイズしてください the `org-odt-fontify-srcblocks` オプションを

12.12.12 Advanced topics in ODT export

The ODT export back-end has extensive features useful 役に立つ for power users and frequent uses of ODT formats.

Configuring a document converter

The ODT export back-end works with popular converters with little or no extra configuration. See 節 12.12.4 [Extending ODT export], ページ 178. The following is for unsupported converters or tweaking existing defaults.

Register the converter
: Add the name of the converter to the `org-odt-convert-processes` variable. Note 注意してください that it also requires how the converter is invoked on the command line. See the variable's docstring for details. 詳細については

Configure its capabilities
: Specify which formats the converter can handle by customizing カスタマイズすることによって the 変数`org-odt-convert-capabilities` を Use 使ってください the entry for the default values in this variable for configuring the new converter. Also see 見てください its docstring そのドキュメント文字列も for details. 詳細については

Choose the converter
: Select the newly added converter as the preferred one by customizing カスタマイズすることによって the オプション`org-odt-convert-process` を

Working with OpenDocument style files

This section explores the internals of the ODT exporter; the means by which it produces styled documents; the use of automatic and custom OpenDocument styles.

The ODT exporter relies on two files for generating its output. These files are bundled with the distribution under the directory pointed to by the variable `org-odt-styles-dir`. The two files are:

'`OrgOdtStyles.xml`'
: This file contributes to the '`styles.xml`' file of the final ODT document. This file gets modified for the following purposes:

  1. To control outline numbering based on user settings;
  2. To add styles generated by '`htmlfontify.el`' for fontification of code blocks.

'`OrgOdtContentTemplate.xml`'
: This file contributes to the '`content.xml`' file of the final ODT document. The contents of the Org outline are inserted between the '`<office:text>`' ... '`</office:text>`' elements of this file.

  Apart from serving as a template file for the final '`content.xml`', the file serves the following purposes:

  1. It contains automatic styles for formatting of tables which are referenced by the exporter;
  2. It contains '`<text:sequence-decl>`' ... '`</text:sequence-decl>`' elements that control numbering of tables, images, equations, and similar entities.

The following two variables control the location from where the ODT exporter picks up the custom styles and content template files. Customize カスタマイズしてください these variables to override the factory styles used by the exporter.

`org-odt-styles-file`
: The ODT export back-end uses the file pointed to by this variable, such as '`styles.xml`', for the final output. It can take one of the following values:

  '`FILE.xml`'
  : Use 使います this file instead かわりに of the default '`styles.xml`'

  '`FILE.odt`' or '`FILE.ott`'
  : Use 使います the '`styles.xml`' contained in the specified OpenDocument Text or Template file

  '`FILE.odt`' or '`FILE.ott`' and a subset of included files
  : Use 使いまう the '`styles.xml`' contained in the specified OpenDocument Text or Template file. Additionally extract the specified member files and embed those within the final ODT document.

    Use 使ってください this option if the '`styles.xml`' file references additional files like header and footer images.

  `nil`
  : Use 使いまづ the default '`styles.xml`'.

`org-odt-content-template-file`
: Use 使いまう this variable to specify the blank '`content.xml`' used in the final output.

Creating one-off styles

The ODT export back-end can read embedded raw OpenDocument XML from the Org file. Such direct formatting is useful 役に立ちます for one-off instances.

Embedding ODT tags as part of regular text
: Enclose OpenDocument syntax in '`@@odt:...@@`' for inline markup. For example, 例えば to highlight a region of text do the following:

```
@@odt:<text:span text:style-name="Highlight">This is highlighted
text</text:span>@@.  But this is regular text.
```

Hint: To see the above example in action, edit the '`styles.xml`' ([Factory styles], ページ 184, を見てください) and add a custom *Highlight* style as shown below:

```
<style:style style:name="Highlight" style:family="text">
  <style:text-properties fo:background-color="#ff0000"/>
</style:style>
```

Embedding a one-line OpenDocument XML
: The ODT export back-end can read one-liner options with '`#+ODT:`' in the Org file. For example, 例えば to force a page break:

```
#+ODT: <text:p text:style-name="PageBreak"/>
```

Hint: To see the above example in action, edit your '`styles.xml`' ([Factory styles], ページ 184, を見てください) and add a custom '`PageBreak`' style as shown below.

```
<style:style style:name="PageBreak" style:family="paragraph"
             style:parent-style-name="Text_20_body">
  <style:paragraph-properties fo:break-before="page"/>
</style:style>
```

Embedding a block of OpenDocument XML
: The ODT export back-end can also read ODT export blocks for OpenDocument XML. Such blocks use the '`#+BEGIN_EXPORT odt`' ... '`#+END_EXPORT`' constructs.

  For example, 例えば to create a one-off paragraph that uses bold text, do the following:

```
#+BEGIN_EXPORT odt
  <text:p text:style-name="Text_20_body_20_bold">
  This paragraph is specially formatted and uses bold text.
  </text:p>
#+END_EXPORT
```

Customizing tables in ODT export

Override the default table format by specifying 指定することによって a custom table style カスタムの表のスタイルを with the '`#+ATTR_ODT`' line. For a discussion 議論については on default formatting of tables, see 節 12.12.7 [Tables in ODT export], ページ 179.

This feature closely mimics the way table templates are defined in the OpenDocument-v1.2 specification[13].

For quick preview of this feature, install the settings below and export the table that follows:

```
(setq org-export-odt-table-styles
      (append org-export-odt-table-styles
              '(("TableWithHeaderRowAndColumn" "Custom"
                 ((use-first-row-styles . t)
                  (use-first-column-styles . t)))
                ("TableWithFirstRowandLastRow" "Custom"
                 ((use-first-row-styles . t)
                  (use-last-row-styles . t))))))
#+ATTR_ODT: :style TableWithHeaderRowAndColumn
| Name  | Phone | Age |
| Peter |  1234 |  17 |
| Anna  |  4321 |  25 |
```

The example above used '`Custom`' template and installed two table styles '`TableWithHeaderRowAndColumn`' and '`TableWithFirstRowandLastRow`'. **Important:** The OpenDocument styles needed for producing the above template were pre-defined. They are available in the section marked '`Custom Table Template`' in '`OrgOdtContentTemplate.xml`'（[Factory styles], ページ 184, を見てください）。For adding new templates, define new styles there.

To use this feature proceed as follows:

1. Create 作成します a table template[14].

   A table template is set of '`table-cell`' and '`paragraph`' styles for each of the following table cell categories:

   - Body
   - First column
   - Last column
   - First row
   - Last row
   - Even row
   - Odd row
   - Even column
   - Odd Column

   The names for the above styles must be chosen based on the name of the table template using a well-defined convention.

   The naming convention is better illustrated with an example. For a table template with the name '`Custom`', the needed style names are listed in the following table.

[13] OpenDocument-v1.2 Specification (`http://docs.oasis-open.org/office/v1.2/OpenDocument-v1.2.html`) を見てください。

[14] See the '`<table:table-template>`' 要素をを見てください。of the OpenDocument-v1.2 使用の

Cell type	Cell style	Paragraph style
Body	'CustomTableCell'	'CustomTableParagraph'
First column	'CustomFirstColumnTableCell'	'CustomFirstColumnTableParagraph'
Last column	'CustomLastColumnTableCell'	'CustomLastColumnTableParagraph'
First row	'CustomFirstRowTableCell'	'CustomFirstRowTableParagraph'
Last row	'CustomLastRowTableCell'	'CustomLastRowTableParagraph'
Even row	'CustomEvenRowTableCell'	'CustomEvenRowTableParagraph'
Odd row	'CustomOddRowTableCell'	'CustomOddRowTableParagraph'
Even column	'CustomEvenColumnTableCell'	'CustomEvenColumnTableParagraph'
Odd column	'CustomOddColumnTableCell'	'CustomOddColumnTableParagraph'

To create 作成するには a table template with the name 'Custom', define the above styles in the '<office:automatic-styles>' ... '</office:automatic-styles>' element of the content template file（[Factory styles], ページ 184, を見てください）。

2. Define a table style[15].

   To define a table style, create an entry for the style in the variable `org-odt-table-styles` and specify the following:

   - the name of the table template created in step (1),
   - the set of cell styles in that template that are to be activated.

   For example, 例えば the entry below defines two different table styles 'TableWithHeaderRowAndColumn' and 'TableWithFirstRowandLastRow' based on the same template 'Custom'. The styles achieve their intended effect by selectively activating the individual cell styles in that template.

```
(setq org-export-odt-table-styles
      (append org-export-odt-table-styles
              '(("TableWithHeaderRowAndColumn" "Custom"
                 ((use-first-row-styles . t)
                  (use-first-column-styles . t)))
                ("TableWithFirstRowandLastRow" "Custom"
                 ((use-first-row-styles . t)
                  (use-last-row-styles . t))))))
```

3. Associate a table with the table style.

   To do this, specify the table style created in step (2) as part of the 'ATTR_ODT' line as shown below.

```
#+ATTR_ODT: :style TableWithHeaderRowAndColumn
| Name  | Phone | Age |
| Peter |  1234 |  17 |
| Anna  |  4321 |  25 |
```

[15] See the attributes 'table:template-name', 'table:use-first-row-styles', 'table:use-last-row-styles', 'table:use-first-column-styles', 'table:use-last-column-styles', 'table:use-banding-rows-styles', and 'table:use-banding-column-styles' of the '<table:table>' element in the OpenDocument-v1.2 specification.

Validating OpenDocument XML

Sometimes ときどき ODT format files may not open due to '`.odt`' file corruption. To verify if such a file is corrupt, validate it against the OpenDocument Relax NG Compact (RNC) syntax schema. But しかし first the '`.odt`' files have to be decompressed using '`zip`' を使って Note 注意してください that '`.odt`' files are ZIP archives: 節 "File Archives" in `emacs`. The contents of ODT files are in XML. For general help with validation---and schema-sensitive editing---of XML files: 節 "Introduction" in `nxml-mode`.

Customize カスタマイズしてください`org-odt-schema-dir` を to point to a directory with OpenDocument RNC files and the needed schema-locating rules. The ODT export back-end takes care of updating the `rng-schema-locating-files`.

12.13 Org Export

org export back-end creates 作成します a normalized version of the Org document in current buffer. The exporter evaluates Babel code（節 14.4 [Evaluating Code Blocks], ページ 221, を見てください）and removes content specific to other back-ends.

Org export commands

`C-c C-e O o` (`org-org-export-to-org`)
: Export エクスポートします。as an Org ファイルとて with a '`.org`' という拡張子をおつ For '`myfile.org`' に対して、Org Org-mode は exports to '`myfile.org.org`' へエクスポートし, overwriting without warning. 警告なしで上書きします。

`C-c C-e O v` (`~~`)
: Export エクスポートします to an Org file, Org-mode のファイルへ then open it. それを開きまづ

12.14 Texinfo Export

12.14.1 Texinfo export commands

`C-c C-e i t` (`org-texinfo-export-to-texinfo`)
: Export エクスポートします as a Texinfo ファイルとして with '`.texi`' という拡張子をもつ For '`myfile.org`' に対して、Org Org-mode は exports to '`myfile.texi`' へエクスポートし, overwriting without warning. 警告なしで上書きします。

`C-c C-e i i` (`org-texinfo-export-to-info`)
: Export エクスポートします to Texinfo フォーマットへ first 最初に and then process 処理します it それを to make an Info file. Info ファイルを To generate 生成すらには other formats, 別のフォーマットを such as DocBook といった customize カスタマイズしてください the `org-texinfo-info-process` 変数を

12.14.2 Texinfo specific export settings

The Texinfo export back-end has several additional keywords for customizing Texinfo output. Setting these keywords これらのキーワードの設定は works similar 同じように動きます to the general options（節 12.2 [Export Settings], ページ 146, を見てください）。

‘SUBTITLE’
: The document subtitle.

‘SUBAUTHOR’
: Additional authors for the document.

‘TEXINFO_FILENAME’
: The Texinfo filename.

‘TEXINFO_CLASS’
: The default document class (`org-texinfo-default-class`), which これは must be a member of `org-texinfo-classes`.

‘TEXINFO_HEADER’
: Arbitrary lines inserted at the end of the header.

‘TEXINFO_POST_HEADER’
: Arbitrary lines inserted after the end of the header.

‘TEXINFO_DIR_CATEGORY’
: The directory category of the document.

‘TEXINFO_DIR_TITLE’
: The directory title of the document.

‘TEXINFO_DIR_DESC’
: The directory description of the document.

‘TEXINFO_PRINTED_TITLE’
: The printed title of the document.

12.14.3 Texinfo file header

After creating the header for a Texinfo file, the Texinfo back-end automatically 自動的に generates 生成します a name and destination path for the Info file. To override this default with a more sensible path and name, specify the ‘TEXINFO_FILENAME’ keyword.

Along with ともに the output's file name, 出力のファイル名と the Texinfo header also contains language details 言語の詳細も（節 12.2 [Export Settings], ページ 146, を見てください）and encoding system as set in the `org-texinfo-coding-system` variable. Insert ‘TEXINFO_HEADER’ keywords for each additional command in the header, for example: 例えば

```
#+TEXINFO_HEADER: @synindex
```

Instead かわりに of repeatedly installing 繰り返しインストールする the same set of commands, define a class in `org-texinfo-classes` once, and then activate it in the document by setting 設定することによって the ‘TEXINFO_CLASS’ keyword to that class.

12.14.4 Texinfo title and copyright page

The default template for hard copy output has a title page with ‘TITLE’ and ‘AUTHOR’ keywords（節 12.2 [Export Settings], ページ 146, を見てください）。To replace the regular title with something different for the printed version, use the ‘TEXINFO_PRINTED_TITLE’ and ‘SUBTITLE’ keywords. Both expect raw Texinfo code for setting their values.

If one ‘AUTHOR’ line is not sufficient, add multiple ‘SUBAUTHOR’ keywords. They have to be set in raw Texinfo code.

```
#+AUTHOR: Jane Smith
#+SUBAUTHOR: John Doe
#+TEXINFO_PRINTED_TITLE: This Long Title@@inlinefmt{tex,@*} Is Broken in @TeX{}
```

Copying material is defined in a dedicated headline with -nil ではない‘COPYING’ プロパティを持つ The back-end inserts 挿入します the contents within a ‘@copying’ command at the beginning of the document. The heading itself does not appear in the structure of the document.

Copyright information is printed on the back of the title page.

```
* Legalese
  :PROPERTIES:
  :COPYING: t
  :END:

  This is a short example of a complete Texinfo file, version 1.0.

  Copyright \copy 2016 Free Software Foundation, Inc.
```

12.14.5 Info directory file

The end result of the Texinfo export process is the creation of an Info file. This Info file’s metadata has variables for category, title, and description: ‘TEXINFO_DIR_CATEGORY’, ‘TEXINFO_DIR_TITLE’, and ‘TEXINFO_DIR_DESC’ keywords that establish where in the Info hierarchy the file fits.

Here これは is an example 例です that writes to the Info directory file:

```
#+TEXINFO_DIR_CATEGORY: Emacs
#+TEXINFO_DIR_TITLE: Org Mode: (org)
#+TEXINFO_DIR_DESC: Outline-based notes management and organizer
```

12.14.6 Headings and sectioning structure

The Texinfo export back-end uses a pre-defined scheme to convert Org headlines to equivalent Texinfo structuring commands. A scheme like this maps top-level headlines to numbered chapters tagged as @chapter and lower-level headlines to unnumbered chapters tagged as @unnumbered. To override such mappings to introduce @part or other Texinfo structuring commands, define a new class in org-texinfo-classes. Activate the new class with the ‘TEXINFO_CLASS’ keyword. When no new class is defined and activated, the Texinfo export back-end defaults to the org-texinfo-default-class.

If an Org headline’s level has no associated Texinfo structuring command, or is below a certain threshold（節 12.2 [Export Settings], ページ 146, を見てください）、then the Texinfo export back-end makes it それを into a list item.

The Texinfo export back-end makes any headline with a non-nil ではない‘APPENDIX’ プロパティを持つ into an appendix. This happens independent of the Org headline level or the ‘TEXINFO_CLASS’ keyword.

The Texinfo export back-end creates 作成します a menu entry after the Org headline for each regular sectioning structure. To override this with a shorter menu entry, use the

‘ALT_TITLE’ property（節 12.3 [Table of Contents], ページ 150, を見てください）。Texinfo menu entries also have an option for a longer ‘DESCRIPTION’ property. Here’s an example that uses both to override the default menu entry:

```
* Controlling Screen Display
  :PROPERTIES:
  :ALT_TITLE: Display
  :DESCRIPTION: Controlling Screen Display
  :END:
```

The text before the first headline belongs to the *Top* node, i.e., すなわち the node in which a reader enters an Info manual. As such, it is expected not to appear in printed output generated from the ‘.texi’ file. See 節 “The Top Node” in texinfo, for more information.

12.14.7 Indices

The Texinfo export back-end recognizes these indexing keywords if used in the Org file: ‘CINDEX’, ‘FINDEX’, ‘KINDEX’, ‘PINDEX’, ‘TINDEX’ and ‘VINDEX’. Write their value as verbatim Texinfo code; in particular, 特に‘{’, ‘}’ and ‘@’ characters need to be escaped エスケープされる必用があります with ‘@’ if they do not belong to a Texinfo command.

```
#+CINDEX: Defining indexing entries
```

For the back-end to generate an index entry for a headline, set the ‘INDEX’ property to ‘cp’ or ‘vr’. These abbreviations come from Texinfo that stand for concept index and variable index. The Texinfo manual has abbreviations for all other kinds of indexes. The back-end exports エクスポートします the headline as an unnumbered chapter or section command, and then inserts 挿入します the index 索引を after its contents.

```
* Concept Index
  :PROPERTIES:
  :INDEX: cp
  :END:
```

12.14.8 Quoting Texinfo code

Use 使ってください any of the following three methods to insert or escape raw Texinfo code:

```
Richard @@texinfo:@sc{@@Stallman@@texinfo:}@@ commence' GNU.

#+TEXINFO: @need800
This paragraph is preceded by...

#+BEGIN_EXPORT texinfo
  @auindex Johnson, Mark
  @auindex Lakoff, George
#+END_EXPORT
```

12.14.9 Plain lists in Texinfo export

The Texinfo export back-end by default デフォルトで converts 変換します description lists in the Org file using 使って the default command ‘@table’, which これは results in 結果になります a table with two columns. To change 変更するには this behavior, このふるまい

を specify '`:table-type`' with '`ftable`' or '`vtable`' attributes. For more information, see 節 "Two-column Tables" in `texinfo`.

The Texinfo export back-end by default デフォルトで also applies a text highlight based on the defaults stored in `org-texinfo-table-default-markup`. To override the default highlight command, specify another one with the '`:indic`' attribute.

Org syntax is limited to one entry per list item. Nevertheless, the Texinfo export back-end can split that entry according 従って to any text provided through the '`:sep`' attribute. Each part then becomes a new entry in the first column of the table.

The following example illustrates all the attributes above:

```
#+ATTR_TEXINFO: :table-type vtable :sep , :indic asis
- foo, bar :: This is the common text for variables foo and bar.
```

becomes

```
@vtable @asis
@item foo
@itemx bar
This is the common text for variables foo and bar.
@end table
```

12.14.10 Tables in Texinfo export

When exporting tables, 表示をエクスポートするときには、the Texinfo エクスポートバックエンドは uses 使います the widest cell width in each column. To override this これを上書きして and instead かわりに specify as fractions of line length, use the '`:columns`' attribute. See example below.

```
#+ATTR_TEXINFO: :columns .5 .5
| a cell | another cell |
```

12.14.11 Images in Texinfo export

Insert 挿入します a file link to the image in the Org file, and the Texinfo export back-end inserts 挿入します the image. そのイメージを These links must have the usual supported image extensions and no descriptions. To scale the image, use '`:width`' and '`:height`' attributes. For alternate text, use '`:alt`' and specify the text using Texinfo code, as shown in the example:

```
#+ATTR_TEXINFO: :width 1in :alt Alternate @i{text}
[[ridt.pdf]]
```

12.14.12 Quotations in Texinfo export

You can write 書くことができます the text of a quotation within a quote block（節 11.1 [Paragraphs], ページ 137, を見てください）。You may also emphasize 強調することもできます some text at the beginning of the quotation with the '`:tag`' attribute.

```
#+ATTR_TEXINFO: :tag Warning
#+BEGIN_QUOTE
Striking your thumb with a hammer may cause severe pain and discomfort.
#+END_QUOTE
```

To specify the author of the quotation, use the '`:author`' attribute.

```
#+ATTR_TEXINFO: :author King Arthur
```

```
#+BEGIN_QUOTE
The Lady of the Lake, her arm clad in the purest shimmering samite,
held aloft Excalibur from the bosom of the water, signifying by divine
providence that I, Arthur, was to carry Excalibur. That is why I am
your king.
#+END_QUOTE
```

12.14.13 Special blocks in Texinfo export

The Texinfo export back-end converts 変換します special blocks スペシャルブロックを to commands with the same name. It also adds any '`:options`' attributes to the end of the command, as shown in this example:

```
#+ATTR_TEXINFO: :options org-org-export-to-org ...
#+BEGIN_defun
  A somewhat obsessive function name.
#+END_defun
```

becomes

```
@defun org-org-export-to-org ...
  A somewhat obsessive function name.
@end defun
```

12.14.14 A Texinfo example

Here これは is a more detailed example Org file. See 節 "GNU Sample Texts" in `texinfo` を見てください for an equivalent example using Texinfo code.

```
#+TITLE: GNU Sample {{{version}}}
#+SUBTITLE: for version {{{version}}}, {{{updated}}}
#+AUTHOR: A.U. Thor
#+EMAIL: bug-sample@gnu.org

#+OPTIONS: ':t toc:t author:t email:t
#+LANGUAGE: en

#+MACRO: version 2.0
#+MACRO: updated last updated 4 March 2014

#+TEXINFO_FILENAME: sample.info
#+TEXINFO_HEADER: @syncodeindex pg cp

#+TEXINFO_DIR_CATEGORY: Texinfo documentation system
#+TEXINFO_DIR_TITLE: sample: (sample)
#+TEXINFO_DIR_DESC: Invoking sample

#+TEXINFO_PRINTED_TITLE: GNU Sample

This manual is for GNU Sample (version {{{version}}},
{{{updated}}}).
```

```
* Copying
  :PROPERTIES:
  :COPYING:  t
  :END:

  This manual is for GNU Sample (version {{{version}}},
  {{{updated}}}),
  which これは
  is an example 例です
  in the Texinfo documentation.

  Copyright \copy 2016 Free Software Foundation, Inc.

  #+BEGIN_QUOTE
  Permission is granted to copy, distribute and/or modify this
  document under the terms of the GNU Free Documentation License,
  Version 1.3 or any later version published by the Free Software
  Foundation; with no Invariant Sections, with no Front-Cover Texts,
  and with no Back-Cover Texts.  A copy of the license is included in
  the section entitled "GNU Free Documentation License".
  #+END_QUOTE

* Invoking sample

  #+PINDEX: sample
  #+CINDEX: invoking @command{sample}

  This is a sample manual.  There is no sample program to invoke, but
  if there were, you could see its basic usage and command line
  options here.

* GNU Free Documentation License
  :PROPERTIES:
  :APPENDIX: t
  :END:

  #+TEXINFO: @include fdl.texi

* Index
  :PROPERTIES:
  :INDEX:    cp
  :END:
```

12.15 iCalendar Export

A large part of Org-mode's interoperability success is its ability to easily export to or import from external applications. The iCalendar export back-end takes calendar data from Org files and exports エクスポートします to the standard iCalendar format.

The iCalendar export back-end can also incorporate TODO entries based on the configuration of the `org-icalendar-include-todo` 変数の The back-end exports エクスポートします plain timestamps as '`VEVENT`', TODO items as '`VTODO`', and also create events from deadlines that are in non-TODO items. The back-end uses the deadlines and scheduling dates in Org TODO items for setting the start and due dates for the iCalendar TODO entry. Consult the `org-icalendar-use-deadline` and `org-icalendar-use-scheduled` variables for more details.

For tags on the headline, the iCalendar export back-end makes them into iCalendar categories. To tweak the inheritance of tags and TODO states, configure 設定してください the variable `org-icalendar-categories`. To assign clock alarms based on time, configure 設定してください the `org-icalendar-alarm-time` variable.

The iCalendar format standard requires globally unique identifier ---or UID ---for each entry. The iCalendar export back-end creates 作成します UID を during export. To save a copy of the UID in the Org file set the variable `org-icalendar-store-UID`. The back-end looks for the '`ID`' property of the entry for re-using the same UID for subsequent exports.

Since a single Org entry can result in multiple iCalendar entries ---timestamp, deadline, scheduled item, or TODO item ---Org adds 追加します prefixes to the UID, depending 依存して on which part of the Org entry triggered the creation of the iCalendar entry. Prefixing ensures UIDs remains unique, yet enable synchronization programs trace the connections.

C-c C-e c f (`org-icalendar-export-to-ics`)
: Create 作成します iCalendar entries from the current Org buffer and store them in the same directory, using a file extension '`.ics`'.

C-c C-e c a (`org-icalendar-export-agenda-files`)
: Create 作成します iCalendar entries from Org files in `org-agenda-files` and store in a separate iCalendar file for each Org file.

C-c C-e c c (`org-icalendar-combine-agenda-files`)
: Create 作成します a combined iCalendar file from Org files in `org-agenda-files` and write it to `org-icalendar-combined-agenda-file` file name.

The iCalendar export back-end includes '`SUMMARY`', '`DESCRIPTION`', '`LOCATION`', '`TIMEZONE`' and '`CLASS`' properties from the Org entries when exporting. エクスポートするときに To force the back-end to inherit the '`LOCATION`', '`TIMEZONE`' and '`CLASS`' properties, configure 設定してください the `org-use-property-inheritance` variable.

When Org entries do not have '`SUMMARY`', '`DESCRIPTION`', '`LOCATION`' and '`CLASS`' properties, the iCalendar export back-end derives the summary from the headline, and derives the description from the body of the Org item. The `org-icalendar-include-body` variable limits the maximum number of characters of the content are turned into its description.

The '`TIMEZONE`' プロパティを can be used 使うことができます to specify 指定するために a per-entry time zone, エントリーごとのタイムゾーンを and is applied 適用されます to any entry with timestamp information. タイムスタンプ情報を持つ Time zones タ

イムゾーンは should be specified as per the IANA time zone database format, e.g., 例えば‘`Asia/Almaty`’. Alternately, あるいは the property value can be ‘`UTC`’, to force UTC time for this entry only.

The ‘`CLASS`’ プロパティを can be used 使うことができます to specify a per-entry visibility class or access restrictions, and is applied to any entry with class information. The iCalendar standard defines three visibility classes:

‘`PUBLIC`’ The entry is publicly visible (this is the default).

‘`CONFIDENTIAL`’
Only a limited group of clients get access to the event.

‘`PRIVATE`’ The entry can be retrieved only by its owner.

The server should treat unknown class properties the same as ‘`PRIVATE`’.

Exporting to iCalendar format depends in large part on the capabilities of the destination application. Some are more lenient than others. Consult the Org-mode FAQ for advice on specific applications.

12.16 Other Built-in Back-ends

Other export back-ends included with Org are:

- ‘`ox-man.el`’: Export エクスポートします。to a man ページへ

To activate such back-ends, either customize カスタマイズしてくださいorg-export-backends を or load directly 直接 with ‘`(require 'ox-man)`’. On successful load, the back-end adds new keys in the export dispatcher（節 12.1 [The Export Dispatcher], ページ 145, を見てください）。

Follow the comment section of such files, for example, ‘`ox-man.el`’, 例えば for usage and configuration details.

12.17 Advanced Export Configuration

Hooks

The export process executes two hooks before the actual exporting begins. The first hook, `org-export-before-processing-hook`, runs before any expansions of macros, Babel code, and include keywords in the buffer. The second hook, `org-export-before-parsing-hook`, runs before the buffer is parsed.

Functions added to these hooks are called with a single argument: the export back-end actually used, as a symbol. You may use them for heavy duty structural modifications of the document. For example, 例えば you can remove every headline in the buffer during export like this: このようにして

```
(defun my-headline-removal (backend)
  "Remove all headlines in the current buffer.
BACKEND is the export back-end being used, as a symbol."
  (org-map-entries
   (lambda () (delete-region (point) (line-beginning-position 2)))))

(add-hook 'org-export-before-parsing-hook 'my-headline-removal)
```

Filters

Filters are lists of functions to be applied to certain parts for a given back-end. The output from the first function in the filter is passed on to the next function in the filter. The final output is the output from the final function in the filter.

The Org export process has many filter sets applicable to different types of objects, plain text, parse trees, export options, and final output formats. The filters are named after the element type or object type: `org-export-filter-TYPE-functions`, where *TYPE* is the type targeted by the filter. Valid types are:

body	bold	babel-call
center-block	clock	code
diary-sexp	drawer	dynamic-block
entity	example-block	export-block
export-snippet	final-output	fixed-width
footnote-definition	footnote-reference	headline
horizontal-rule	inline-babel-call	inline-src-block
inlinetask	italic	item
keyword	latex-environment	latex-fragment
line-break	link	node-property
options	paragraph	parse-tree
plain-list	plain-text	planning
property-drawer	quote-block	radio-target
section	special-block	src-block
statistics-cookie	strike-through	subscript
superscript	table	table-cell
table-row	target	timestamp
underline	verbatim	verse-block

Here これは is an example filter that replaces non-breaking spaces in the Org buffer with '~' for the LaTeX back-end.

```
(defun my-latex-filter-nobreaks (text backend info)
  "Ensure \" \" are properly handled in LaTeX export."
  (when (org-export-derived-backend-p backend 'latex)
    (replace-regexp-in-string " " "~" text)))

(add-to-list 'org-export-filter-plain-text-functions
             'my-latex-filter-nobreaks)
```

A filter requires three arguments: the code to be transformed, the name of the back-end, and some optional information about the export process. The third argument can be safely ignored. Note 注意してください the use 使用に of `org-export-derived-backend-p` predicate that tests for *latex* back-end or any other back-end, such as *beamer* といった derived from *latex*.

Defining filters for individual files

The Org export can filter not just for back-ends, but also for specific files through the '`BIND`' keyword. Here これは is an example 例です with two filters; one removes brackets from time stamps, and the other removes strike-through text. The filter functions are defined in

a code block in the same Org file, which これが is a handy location 便利な場所です for debugging. デバッグ用に

```
#+BIND: org-export-filter-timestamp-functions (tmp-f-timestamp)
#+BIND: org-export-filter-strike-through-functions (tmp-f-strike-through)
#+BEGIN_SRC emacs-lisp :exports results :results none
  (defun tmp-f-timestamp (s backend info)
    (replace-regexp-in-string "&[lg]t;\\|[][]" "" s))
  (defun tmp-f-strike-through (s backend info) "")
#+END_SRC
```

Extending an existing back-end

Some parts of the conversion process can be extended for certain elements so as to introduce a new or revised translation. That is how the HTML export back-end was extended to handle Markdown format. The extensions work seamlessly so any aspect of filtering not done by the extended back-end is handled by the original back-end. Of all the export customization in Org, extending is very powerful as it operates at the parser level.

For this example, この例に対して make the *ascii* back-end display the language used in a source code block. Also また make it それに display only when some attribute is non-`nil`, `nil` でない値のときに like the following: 次のようにして

```
#+ATTR_ASCII: :language t
```

Then extend ASCII back-end with a custom "my-ascii" back-end.

```
(defun my-ascii-src-block (src-block contents info)
  "Transcode a SRC-BLOCK element from Org to ASCII.
CONTENTS is nil.  INFO is a plist used as a communication
channel."
  (if (not (org-export-read-attribute :attr_ascii src-block :language))
      (org-export-with-backend 'ascii src-block contents info)
    (concat
     (format ",--[ %s ]--\n%s`----"
             (org-element-property :language src-block)
             (replace-regexp-in-string
              "^" "| "
              (org-element-normalize-string
               (org-export-format-code-default src-block info)))))))

(org-export-define-derived-backend 'my-ascii 'ascii
  :translate-alist '((src-block . my-ascii-src-block)))
```

The `my-ascii-src-block` function looks at the attribute above the current element. If not true, hands over to *ascii* back-end. If true, which it is in this example, この例の中では it それは creates 作成します a box 箱を around the code and leaves room for the inserting a string for language. The last form creates 作成します the new back-end that springs to action only when translating `src-block` type elements.

To use the newly defined back-end, evaluate the following from an Org buffer:

```
(org-export-to-buffer 'my-ascii "*Org MY-ASCII Export*")
```

Further steps to consider would be an interactive function, self-installing an item in the export dispatcher menu, and other user-friendly improvements.

12.18 Export in Foreign Buffers

The export back-ends in Org often include commands to convert 変換します selected regions. A convenient feature of this in-place conversion is that the exported output replaces the original source. Here これらは are such functions:

`org-ascii-convert-region-to-ascii`
: Convert the selected region into ASCII.

`org-ascii-convert-region-to-utf8`
: Convert the selected region into UTF-8.

`org-html-convert-region-to-html`
: Convert the selected region into HTML.

`org-latex-convert-region-to-latex`
: Convert the selected region into LaTeX.

`org-texinfo-convert-region-to-texinfo`
: Convert the selected region into Texinfo.

`org-md-convert-region-to-md`
: Convert the selected region into Markdown.

In-place conversions are particularly handy for quick conversion of tables and lists in foreign buffers. For example, 例えば in an HTML buffer, write a list in Org syntax, select it, and convert it to HTML with *`M-x org-html-convert-region-to-html`*.

13 Publishing

Org includes a publishing management system that allows you to configure 設定することを automatic HTML conversion of *projects* composed of interlinked Org files. You can also configure 設定することもできまう Org Org-mode を to automatically 自動的に upload your exported HTML pages and related attachments, such as images イメージや and source code files, ソースコードファイルといっあ to a web server. ウエブサーバーへ

You can also use Org to convert files into PDF, or even combine HTML and PDF conversion so that files are available in both formats on the server.

Publishing has been contributed to Org by David O'Toole.

13.1 Configuration

Publishing needs 必用ですう significant configuration to specify files, destination and many other properties of a project.

13.1.1 The variable `org-publish-project-alist`

Publishing is configured almost entirely through setting the value of one variable, called `org-publish-project-alist`. Each element of the list configures one project, and may be in one of the two following forms:

```
("project-name" :property value :property value ...)
```

i.e., すなわち a well-formed property list with alternating keys and values, or:

```
("project-name" :components ("project-name" "project-name" ...))
```

In both cases, projects are configured by specifying 指定することによって property values. プロパティ値を A project defines the set of files that are to be published, as well as the publishing configuration to use when publishing those files. When a project takes the second form listed above, the individual members of the `:components` property are taken to be sub-projects, which group together files requiring different publishing options. When you publish such a "meta-project", all the components are also published, in the sequence given.

13.1.2 Sources and destinations for files

Most properties are optional, but some should always 常に be set. In particular, 特に Org Org-mode は needs to know 知る必用がありますwhere to どこを look for 探すべきかと source files, and where どこへ to put published files.

`:base-directory`
: Directory containing 含んでいる publishing source files.

`:publishing-directory`
: Directory where output files are published. You can directly publish to a web-server using a file name syntax appropriate for the Emacs tramp package. Or または you can publish to a local directory and use external tools to upload your website（節 13.2 [Uploading Files], ページ 208, ををを見てください）。

`:preparation-function`
: Function or list of functions to be called before starting the publishing process, for example, 例えば to run '`make`' for updating files to be published. Each preparation function is called with a single argument, the project property list.

`:completion-function`
: Function or list of functions called after finishing the publishing process, for example, 例えば例えば to change permissions of the resulting files. Each completion function is called with a single argument, the project property list.

13.1.3 Selecting files

By default, デフォルトで all files with extension '`.org`' in the base directory are considered part of the project. This can be modified by setting 設定することによって the following properties

`:base-extension`
: Extension---without the dot---of source files. This actually is a regular expression. Set this to the symbol `any` if you want to get all files in `:base-directory`, even without extension.

`:exclude` Regular expression to match file names that should not be published, even though they have been selected on the basis of their extension.

`:include` List of files to be included regardless of `:base-extension` and `:exclude`.

`:recursive`
: Non-`nil` `nil` でない値は means, 意味します check base-directory recursively for files to publish.

13.1.4 Publishing action

Publishing means that a file is copied to the destination directory and possibly transformed in the process. The default transformation デフォルトの変換は is to export Org files as HTML files, and this is done by the function `org-publish-org-to-html` which calls the HTML exporter（節 12.9 [HTML Export], ページ 159, を見てください）。But しかし you can also publish your content as PDF files using `org-publish-org-to-pdf`, or as ASCII, Texinfo, etc., using the corresponding functions.

If you want to publish the Org file as an '`.org`' file but with *archived*, *commented*, and *tag-excluded* trees removed, use `org-publish-org-to-org`. This produces '`file.org`' and put it in the publishing directory. If you want a htmlized version of this file, set the parameter `:htmlized-source` to `t`. It produces '`file.org.html`' in the publishing directory[1].

Other files like images only need to be copied コピーされる必用があるだけです to the publishing destination; for this このためには you can use 使うことができます`org-publish-attachment` を For non-Org files, Org-mode でない ファイルに対しては、you always 常に need to specify 指定する必用があります the publishing function: 出版関数を

[1] If the publishing directory is the same as the source directory, '`file.org`' is exported as '`file.org.org`', so you probably do not want to do this.

`:publishing-function`
: Function executing the publication of a file. This may also be a list of functions, which これらは are all called in turn. 全て順番に呼ばれます

`:htmlized-source`
: Non-`nil` `nil` でない値あ means, 意味しんます publish htmlized source.

The function must accept three arguments: a property list containing 含んでいる at least a `:publishing-directory` property, the name of the file to be published, and the path to the publishing directory of the output file. It should take the specified file, make the necessary transformation, if any, and place the result into the destination folder.

13.1.5 Options for the exporters

The property list プロパティリストを can be used 使うことができます to set 設定するために many export options 多くのエクスポートオプションを for the HTML and LaTeX exporters. In most cases, ほとんどの場合に these properties これらのプロパティは correspond 対応します to user variables ユーザー変数にん in Org. Org-mode の中の The table below 下の表が lists リストしています these properties これらのプロパティを along with the variable 変数とともに they belong to. それらが属している See 見てください the documentation string for the respective variable for details. 詳細については

When a property is given a value in `org-publish-project-alist`, its setting overrides the value of the corresponding user variable, if any, during publishing. Options set within a file（節 12.2 [Export Settings], ページ 146, を見てください）however, しかし override everything. 全てを上書きします

Generic properties

`:archived-trees`	`org-export-with-archived-trees`
`:exclude-tags`	`org-export-exclude-tags`
`:headline-levels`	`org-export-headline-levels`
`:language`	`org-export-default-language`
`:preserve-breaks`	`org-export-preserve-breaks`
`:section-numbers`	`org-export-with-section-numbers`
`:select-tags`	`org-export-select-tags`
`:with-author`	`org-export-with-author`
`:with-broken-links`	`org-export-with-broken-links`
`:with-clocks`	`org-export-with-clocks`
`:with-creator`	`org-export-with-creator`
`:with-date`	`org-export-with-date`
`:with-drawers`	`org-export-with-drawers`
`:with-email`	`org-export-with-email`
`:with-emphasize`	`org-export-with-emphasize`
`:with-fixed-width`	`org-export-with-fixed-width`
`:with-footnotes`	`org-export-with-footnotes`
`:with-latex`	`org-export-with-latex`
`:with-planning`	`org-export-with-planning`
`:with-priority`	`org-export-with-priority`
`:with-properties`	`org-export-with-properties`

:with-special-strings	org-export-with-special-strings
:with-sub-superscript	org-export-with-sub-superscripts
:with-tables	org-export-with-tables
:with-tags	org-export-with-tags
:with-tasks	org-export-with-tasks
:with-timestamps	org-export-with-timestamps
:with-title	org-export-with-title
:with-toc	org-export-with-toc
:with-todo-keywords	org-export-with-todo-keywords

ASCII specific properties

:ascii-bullets	org-ascii-bullets
:ascii-caption-above	org-ascii-caption-above
:ascii-charset	org-ascii-charset
:ascii-global-margin	org-ascii-global-margin
:ascii-format-drawer-function	org-ascii-format-drawer-function
:ascii-format-inlinetask-function	org-ascii-format-inlinetask-function
:ascii-headline-spacing	org-ascii-headline-spacing
:ascii-indented-line-width	org-ascii-indented-line-width
:ascii-inlinetask-width	org-ascii-inlinetask-width
:ascii-inner-margin	org-ascii-inner-margin
:ascii-links-to-notes	org-ascii-links-to-notes
:ascii-list-margin	org-ascii-list-margin
:ascii-paragraph-spacing	org-ascii-paragraph-spacing
:ascii-quote-margin	org-ascii-quote-margin
:ascii-table-keep-all-vertical-lines	org-ascii-table-keep-all-vertical-lines
:ascii-table-use-ascii-art	org-ascii-table-use-ascii-art
:ascii-table-widen-columns	org-ascii-table-widen-columns
:ascii-text-width	org-ascii-text-width
:ascii-underline	org-ascii-underline
:ascii-verbatim-format	org-ascii-verbatim-format

Beamer specific properties

:beamer-theme	org-beamer-theme
:beamer-column-view-format	org-beamer-column-view-format
:beamer-environments-extra	org-beamer-environments-extra
:beamer-frame-default-options	org-beamer-frame-default-options
:beamer-outline-frame-options	org-beamer-outline-frame-options
:beamer-outline-frame-title	org-beamer-outline-frame-title
:beamer-subtitle-format	org-beamer-subtitle-format

HTML specific properties

:html-allow-name-attribute-in-anchors	org-html-allow-name-attribute-in-anchors
:html-checkbox-type	org-html-checkbox-type
:html-container	org-html-container-element
:html-divs	org-html-divs

:html-doctype	org-html-doctype
:html-extension	org-html-extension
:html-footnote-format	org-html-footnote-format
:html-footnote-separator	org-html-footnote-separator
:html-footnotes-section	org-html-footnotes-section
:html-format-drawer-function	org-html-format-drawer-function
:html-format-headline-function	org-html-format-headline-function
:html-format-inlinetask-function	org-html-format-inlinetask-function
:html-head-extra	org-html-head-extra
:html-head-include-default-style	org-html-head-include-default-style
:html-head-include-scripts	org-html-head-include-scripts
:html-head	org-html-head
:html-home/up-format	org-html-home/up-format
:html-html5-fancy	org-html-html5-fancy
:html-indent	org-html-indent
:html-infojs-options	org-html-infojs-options
:html-infojs-template	org-html-infojs-template
:html-inline-image-rules	org-html-inline-image-rules
:html-inline-images	org-html-inline-images
:html-link-home	org-html-link-home
:html-link-org-files-as-html	org-html-link-org-files-as-html
:html-link-up	org-html-link-up
:html-link-use-abs-url	org-html-link-use-abs-url
:html-mathjax-options	org-html-mathjax-options
:html-mathjax-template	org-html-mathjax-template
:html-metadata-timestamp-format	org-html-metadata-timestamp-format
:html-postamble-format	org-html-postamble-format
:html-postamble	org-html-postamble
:html-preamble-format	org-html-preamble-format
:html-preamble	org-html-preamble
:html-self-link-headlines	org-html-self-link-headlines
:html-table-align-individual-field	de{org-html-table-align-individual-fields
:html-table-attributes	org-html-table-default-attributes
:html-table-caption-above	org-html-table-caption-above
:html-table-data-tags	org-html-table-data-tags
:html-table-header-tags	org-html-table-header-tags
:html-table-row-tags	org-html-table-row-tags
:html-table-use-header-tags-for-first-column	org-html-table-use-header-tags-for-first-colu
:html-tag-class-prefix	org-html-tag-class-prefix
:html-text-markup-alist	org-html-text-markup-alist
:html-todo-kwd-class-prefix	org-html-todo-kwd-class-prefix
:html-toplevel-hlevel	org-html-toplevel-hlevel
:html-use-infojs	org-html-use-infojs
:html-validation-link	org-html-validation-link
:html-viewport	org-html-viewport
:html-xml-declaration	org-html-xml-declaration

LaTeX specific properties

:latex-active-timestamp-format	org-latex-active-timestamp-format
:latex-caption-above	org-latex-caption-above
:latex-classes	org-latex-classes
:latex-class	org-latex-default-class
:latex-compiler	org-latex-compiler
:latex-default-figure-position	org-latex-default-figure-position
:latex-default-table-environment	org-latex-default-table-environment
:latex-default-table-mode	org-latex-default-table-mode
:latex-diary-timestamp-format	org-latex-diary-timestamp-format
:latex-footnote-defined-format	org-latex-footnote-defined-format
:latex-footnote-separator	org-latex-footnote-separator
:latex-format-drawer-function	org-latex-format-drawer-function
:latex-format-headline-function	org-latex-format-headline-function
:latex-format-inlinetask-function	org-latex-format-inlinetask-function
:latex-hyperref-template	org-latex-hyperref-template
:latex-image-default-height	org-latex-image-default-height
:latex-image-default-option	org-latex-image-default-option
:latex-image-default-width	org-latex-image-default-width
:latex-images-centered	org-latex-images-centered
:latex-inactive-timestamp-format	org-latex-inactive-timestamp-format
:latex-inline-image-rules	org-latex-inline-image-rules
:latex-link-with-unknown-path-format	org-latex-link-with-unknown-path-format
:latex-listings-langs	org-latex-listings-langs
:latex-listings-options	org-latex-listings-options
:latex-listings	org-latex-listings
:latex-minted-langs	org-latex-minted-langs
:latex-minted-options	org-latex-minted-options
:latex-prefer-user-labels	org-latex-prefer-user-labels
:latex-subtitle-format	org-latex-subtitle-format
:latex-subtitle-separate	org-latex-subtitle-separate
:latex-table-scientific-notation	org-latex-table-scientific-notation
:latex-tables-booktabs	org-latex-tables-booktabs
:latex-tables-centered	org-latex-tables-centered
:latex-text-markup-alist	org-latex-text-markup-alist
:latex-title-command	org-latex-title-command
:latex-toc-command	org-latex-toc-command

Markdown specific properties

:md-footnote-format	org-md-footnote-format
:md-footnotes-section	org-md-footnotes-section
:md-headline-style	org-md-headline-style

ODT specific properties

:odt-content-template-file	org-odt-content-template-file
:odt-display-outline-level	org-odt-display-outline-level

:odt-fontify-srcblocks	org-odt-fontify-srcblocks
:odt-format-drawer-function	org-odt-format-drawer-function
:odt-format-headline-function	org-odt-format-headline-function
:odt-format-inlinetask-function	org-odt-format-inlinetask-function
:odt-inline-formula-rules	org-odt-inline-formula-rules
:odt-inline-image-rules	org-odt-inline-image-rules
:odt-pixels-per-inch	org-odt-pixels-per-inch
:odt-styles-file	org-odt-styles-file
:odt-table-styles	org-odt-table-styles
:odt-use-date-fields	org-odt-use-date-fields

Texinfo specific properties

:texinfo-active-timestamp-format	org-texinfo-active-timestamp-format
:texinfo-classes	org-texinfo-classes
:texinfo-class	org-texinfo-default-class
:texinfo-table-default-markup	org-texinfo-table-default-markup
:texinfo-diary-timestamp-format	org-texinfo-diary-timestamp-format
:texinfo-filename	org-texinfo-filename
:texinfo-format-drawer-function	org-texinfo-format-drawer-function
:texinfo-format-headline-function	org-texinfo-format-headline-function
:texinfo-format-inlinetask-function	org-texinfo-format-inlinetask-function
:texinfo-inactive-timestamp-format	org-texinfo-inactive-timestamp-format
:texinfo-link-with-unknown-path-format	org-texinfo-link-with-unknown-path-format
:texinfo-node-description-column	org-texinfo-node-description-column
:texinfo-table-scientific-notation	org-texinfo-table-scientific-notation
:texinfo-tables-verbatim	org-texinfo-tables-verbatim
:texinfo-text-markup-alist	org-texinfo-text-markup-alist

13.1.6 Publishing links

To create 作成するには a link リンクを from one Org file 1 つの Org-mode のファイルから to another, 別の Org-mode のファイルへの you would use 使います something like '`[[file:foo.org][The foo]]`' or simply 単純に'`[[file:foo.org]]`'（節 4.4 [External Links], ページ 41, を見てください）。When published, this link becomes なります a link to '`foo.html`'. You can thus よって interlink the pages of your "Org web" project and the links will work as expected when you publish them to HTML. If you also publish the Org source file and want to link to it, use an '`http`' link instead かわりに of a '`file:`' リンクの because '`file`' links are converted to link to the corresponding '`.html`' file.

You may also link to related files, such as images. イメージといった Provided you are careful with relative file names, and provided you have also configured Org to upload the related files, these links will work too. See 節 13.3.2 [Complex example], ページ 209, for an example of this usage.

Eventually, links between published documents can contain some search options（節 4.8 [Search Options], ページ 48, を見てください）、which will be resolved to the appropriate location in the linked file. For example, 例えば once published to HTML, the following links all point to a dedicated anchor in '`foo.html`'.

```
[[file:foo.org::*heading]]
```

```
[[file:foo.org::#custom-id]]
[[file:foo.org::target]]
```

13.1.7 Generating a sitemap

The following properties may be used to control publishing of a map of files for a given project.

`:auto-sitemap`
: When non-`nil`, `nil` でない値のときには publish a sitemap during `org-publish-current-project` or `org-publish-all`.

`:sitemap-filename`
: Filename for output of sitemap. Defaults to デフォルトは‘`sitemap.org`’ であり、which これは becomes ‘`sitemap.html`’ になるいます。

`:sitemap-title`
: Title of sitemap page. サイトマップページの名前です。Defaults to name of file. デフォルトはファイルの名前でんえす。

`:sitemap-format-entry`
: With this option one can tell how a site-map entry is formatted in the site-map. It is a function called with three arguments: the file or directory name relative to base directory of the project, the site-map style and the current project. It is expected to return a string. Default value デフォルト値は turns 変え file names into links and use 使います document titles as descriptions. 説明として For specific formatting needs, one can use 使うことができます`org-publish-find-date` と`org-publish-find-title` と`org-publish-find-property` を to retrieve 取り出すために additional information 追加の情報を about published documents. 出版されたドキュメントについての

`:sitemap-function`
: Plug-in function to use for generation of the sitemap. It is called with two arguments: the title of the site-map and a representation of the files and directories involved in the project as a nested list, which これを can further be transformed さらに変換することができまう using つかって`org-list-to-generic`, `org-list-to-subtree` and alike. Default value デフォルト値は generates 生成します a plain list of links to all files in the project.

`:sitemap-sort-folders`
: Where folders should appear in the sitemap. Set this to `first` (default) or `last` to display folders first or last, respectively. When set 設定されているときには、to `ignore` に folders are ignored altogether. 完全に無視されます Any other value mixes files and folders. This variable has no effect when site-map style is `tree`.

`:sitemap-sort-files`
: How the files are sorted in the site map. Set this to `alphabetically` (default), `chronologically` or `anti-chronologically`. `chronologically` sorts the files with older date first while `anti-chronologically` sorts the files with newer date first. `alphabetically` sorts the files alphabetically. The date of a file is retrieved with `org-publish-find-date`.

`:sitemap-ignore-case`
: Should sorting be case-sensitive? デフォルトは`nil` です。

`:sitemap-file-entry-format`
: With this option one can tell how a sitemap's entry is formatted in the sitemap. This is a format string with some escape sequences: `%t` stands for the title of the file, `%a` stands for the author of the file and `%d` stands for the date of the file. The date is retrieved with the `org-publish-find-date` function and formatted with `org-publish-sitemap-date-format`. デフォルトは`%t` です。

`:sitemap-date-format`
: Format string for the `format-time-string` function that tells how a sitemap entry's date is to be formatted. This property bypasses `org-publish-sitemap-date-format` which defaults to `%Y-%m-%d`.

13.1.8 Generating an index

Org-mode can generate an index across the files of a publishing project.

`:makeindex`
: When non-`nil`, `nil` でない値のときにぬうあ generate 生成します in index in the file '`theindex.org`' and publish it as '`theindex.html`'.

The file このファイルは is created 作成されます when first publishing a project with the `:makeindex` set. The file only contains a statement '`#+INCLUDE: "theindex.inc"`'. You can then build around this include statement by adding a title, style information, etc.

Index entries are specified with '`INDEX`' keyword. An entry that contains 含んでいる an exclamation mark creates 作成します a sub item.

```
*** Curriculum Vitae
#+INDEX: CV
#+INDEX: Application!CV
```

13.2 Uploading Files

For those people already utilizing third party sync tools such as Rsync or Unison, it might be preferable not to use the built-in remote publishing facilities of Org-mode which rely heavily on Tramp. Tramp, while very useful とても役に立ち and powerful, tends not to be so efficient for multiple file transfer and has been known to cause problems under heavy usage.

Specialized synchronization utilities offer 提供します several advantages. In addition to timestamp comparison, they also do content and permissions/attribute checks. For this reason you might prefer to publish your web to a local directory---possibly even *in place* with your Org files---and then use Unison or Rsync to do the synchronization with the remote host.

Since Unison, for example, 例えば can be configured 設定することができます as to which files to transfer to a certain remote destination, it can greatly simplify the project publishing definition. Simply 単純に keep all files in the correct location, process your Org files with `org-publish` and let the synchronization tool do the rest. You do not need, in this scenario, to include attachments such as JPG, CSS or PNG files in the project definition since the third-party tool syncs them.

Publishing to a local directory is also much faster than to a remote one, so that you can afford more easily to republish entire projects. If you set `org-publish-use-timestamps-flag` を`nil` に you gain the main benefit of re-including 含む any changed external files such as source example files ソースの例のファイルといった you might include with '`INCLUDE`' keyword. The timestamp mechanism in Org is not smart enough to detect if included files have been modified.

13.3 Sample Configuration

Below we provide two example configurations. The first one is a simple project publishing only a set of Org files. The second example is more complex, with a multi-component project.

13.3.1 Example: simple publishing configuration

This example publishes a set of Org files to the '`public_html`' directory on the local machine.

```
(setq org-publish-project-alist
      '(("org"
         :base-directory "~/org/"
         :publishing-directory "~/public_html"
         :section-numbers nil
         :table-of-contents nil
         :style "<link rel=\"stylesheet\"
                href=\"../other/mystyle.css\"
                type=\"text/css\"/>")))
```

13.3.2 Example: complex publishing configuration

This more complicated example publishes an entire website, including 含む Org files converted to HTML, image files, Emacs Lisp source code, and style sheets. The publishing directory is remote and private files are excluded.

To ensure that links are preserved, care should be taken to replicate your directory structure on the web server, and to use relative file paths. For example, 例えば if your Org files are kept in '`~/org/`' and your publishable images in '`~/images/`', you would link to an image with

```
file:../images/myimage.png
```

On the web server, the relative path to the image should be the same. You can accomplish this by setting up セットアップすることにようおって an '`images/`' folder in the right place on the web server, and publishing images to it.

```
(setq org-publish-project-alist
      '(("orgfiles"
         :base-directory "~/org/"
         :base-extension "org"
         :publishing-directory "/ssh:user@host:~/html/notebook/"
         :publishing-function org-html-publish-to-html
         :exclude "PrivatePage.org" ;; regexp
         :headline-levels 3
```

```
 :section-numbers nil
 :with-toc nil
 :html-head "<link rel=\"stylesheet\"
         href=\"../other/mystyle.css\" type=\"text/css\"/>"
 :html-preamble t)

("images"
 :base-directory "~/images/"
 :base-extension "jpg\\|gif\\|png"
 :publishing-directory "/ssh:user@host:~/html/images/"
 :publishing-function org-publish-attachment)

("other"
 :base-directory "~/other/"
 :base-extension "css\\|el"
 :publishing-directory "/ssh:user@host:~/html/other/"
 :publishing-function org-publish-attachment)
("website" :components ("orgfiles" "images" "other"))))
```

13.4 Triggering Publication

Once properly configured, Org can publish with the following commands:

C-c C-e X (`org-publish`)
: Prompt for a specific project and publish all files that belong to it.

C-c C-e P (`org-publish-current-project`)
: Publish the project プロジェクトを出版します。containing 含んでいる the current file. 現在のファイルを

C-c C-e F (`org-publish-current-file`)
: Publish only the current file.

C-c C-e E (`org-publish-all`)
: Publish every project.

Org uses timestamps to track when a file has changed. The above functions normally 通常は only publish changed files. You can override this これを上書きして、and force 強制することができます publishing of all files by giving a prefix argument to any of the commands above, or by customizing カスタマイズすることによって the 変数`org-publish-use-timestamps-flag` を This may be necessary in particular if files include other files via ‘`SETUPFILE`’ or ‘`INCLUDE`’ keywords.

14 Working with Source Code

Source code here refers to any plain text collection of computer instructions, possibly with comments, written using a human-readable programming language. Org can manage source code in an Org document when the source code is identified with begin and end markers. Working with source code begins with identifying source code blocks. A source code block can be placed almost anywhere in an Org document; it is not restricted to the preamble or the end of the document. However, しかし Org Org-mode は cannot manage 管理することはできません a source code block if it is placed inside an Org comment or within a fixed width section.

Here これは is an example source code block in the Emacs Lisp language:

```
#+BEGIN_SRC emacs-lisp
  (defun org-xor (a b)
     "Exclusive or."
     (if a (not b) b))
#+END_SRC
```

Org can manage the source code in the block delimited by '`#+BEGIN_SRC`' ... '`#+END_SRC`' in several ways that can simplify housekeeping tasks essential to modern source code maintenance. Org can edit, format, extract, export, and publish source code blocks. Org can also compile and execute a source code block, then capture the results. The Org mode literature sometimes ときどき refers 指します to source code blocks as *live code* blocks because they can alter the content of the Org document or the material that it exports. エクスポートします Users can control how live they want each source code block by tweaking the header arguments（節 14.2 [Using Header Arguments], ページ 213, を見てください）for compiling, execution, extraction, and exporting.

Source code blocks are one of many Org block types, which これは also また include "center", "comment", "dynamic", "example", "export", "quote", "special", and "verse". This section pertains to blocks between '`#+BEGIN_SRC`' and '`#+END_SRC`'.

For editing and formatting a source code block, Org uses an appropriate Emacs major mode that includes features specifically designed for source code in that language.

Org can extract one or more source code blocks and write them to one or more source files---a process known as *tangling* in literate programming terminology.

For exporting and publishing, Org's back-ends can format a source code block appropriately, often with native syntax highlighting.

For executing and compiling a source code block, the user can configure 設定することができあますう Org to select the appropriate compiler. Org provides 提供しています facilities to collect the result of the execution or compiler output, insert it into the Org document, and/or export it. In addition 加えて to text results, Org Org-mode は can insert 挿入することができます links リンクを to other data types, including 含む audio, video, and graphics. Org can also link a compiler error message to the appropriate line in the source code block.

An important feature of Org's management of source code blocks is the ability to pass variables, functions, and results to one another 互いに using 使って a common syntax for source code blocks in any language. Although most literate programming facilities are restricted to one language or another, Org's language-agnostic approach lets the literate programmer match each programming task with the appropriate computer language and

to mix them all together in a single Org document. This interoperability among languages explains why Org's source code management facility was named *Org Babel* by its originators, Eric Schulte and Dan Davison.

Org-mode fulfills the promise of easy verification and maintenance of publishing reproducible research by keeping text, data, code, configuration settings of the execution environment, the results of the execution, and associated narratives, claims, references, and internal and external links in a single Org document.

Details of Org's facilities for working with source code are described in the following sections.

14.1 Structure of Code Blocks

Org Org-mode は offers 提供します two ways 2 つの方法を to structure source code in Org documents: in a source code block, and directly inline. Both specifications are shown below.

A source code block conforms to this structure:

```
#+NAME: <name>
#+BEGIN_SRC <language> <switches> <header arguments>
  <body>
#+END_SRC
```

Do not be put-off by having to remember the source block syntax. Org mode Org-mode は offers 提供します a command コマンドを for wrapping existing text in a block（節 15.2 [Structure Templates], ページ 240, を見てください）。Org also works with other completion systems in Emacs, some of which predate Org and have custom domain-specific languages for defining templates. Regular use of templates reduces errors, increases accuracy, and maintains consistency.

An inline code block conforms to this structure:

```
src_<language>{<body>}
```

or

```
src_<language>[<header arguments>]{<body>}
```

'`#+NAME: <name>`'
: Optional. Names the source block so it can be called, like a function, from other source blocks or inline code to evaluate or to capture the results. Code from other blocks, other files, and from table formulas（節 3.5 [The Spreadsheet], ページ 26, を見てください）can use 使うことができます the name この名前を to reference a source block. This naming serves the same purpose as naming Org tables. Org-mode requires unique names. For duplicate names, Org-mode's behavior is undefined.

'`#+BEGIN_SRC`' ... '`#+END_SRC`'
: Mandatory. They mark the start and end of a block that Org requires. The '`#+BEGIN_SRC`' line takes additional arguments, as described next.

'`<language>`'
: Mandatory. It is the identifier of the source code language in the block. See 節 14.8 [Languages], ページ 232, for identifiers of supported languages.

‘<switches>’
: Optional. Switches provide finer control of the code execution, export, and format（節 11.6 [Literal Examples], ページ 141, の中のスイッチの議論を見てください）。

‘<header arguments>’
: Optional. Heading arguments control many aspects of evaluation, export and tangling of code blocks（節 14.2 [Using Header Arguments], ページ 213, を見てください）。Using Org's properties feature, header arguments can be selectively applied to the entire buffer or specific sub-trees of the Org document.

‘<body>’ Source code in the dialect of the specified language identifier.

14.2 Using Header Arguments

Org comes with many header arguments common to all languages. New header arguments are added for specific languages as they become available for use in source code blocks. A header argument is specified with an initial colon followed by the argument's name in lowercase.

Since header arguments can be set in several ways, Org prioritizes them in case of overlaps or conflicts by giving local settings a higher priority. Header values in function calls, for example, 例えば override header values from global defaults.

System-wide header arguments

System-wide values of header arguments can be specified by customizing カスタマイズすることによって the `org-babel-default-header-args` 変数を which これは defaults to the following values:

```
:session    => "none"
:results    => "replace"
:exports    => "code"
:cache      => "no"
:noweb      => "no"
```

The example below sets ‘`:noweb`’ header arguments to ‘`yes`’, which これは makes Org expand ‘`:noweb`’ references by default. デフォルトで

```
(setq org-babel-default-header-args
      (cons '(:noweb . "yes")
            (assq-delete-all :noweb org-babel-default-header-args)))
```

Each language can have separate default header arguments by customizing the variable `org-babel-default-header-args:<LANG>`, where *<LANG>* is the name of the language. For details, 詳細については see 見てください the language-specific online documentation at `https://orgmode.org/worg/org-contrib/babel/` みある

Header arguments in Org-mode properties

For header arguments applicable to the buffer, use ‘`PROPERTY`’ keyword anywhere in the Org file（節 7.1 [Property Syntax], ページ 66, を見てください）。

The following example makes all the R code blocks execute in the same session. Setting '`:results`' to '`silent`' ignores the results of executions for all blocks, not just R code blocks; no results inserted for any block.

```
#+PROPERTY: header-args:R  :session *R*
#+PROPERTY: header-args    :results silent
```

Header arguments set through Org's property drawers（節 7.1 [Property Syntax], ページ 66, を見てください）apply at the sub-tree level on down. Since these property drawers can appear anywhere in the file hierarchy, Org uses outermost call or source block to resolve the values. Org ignores `org-use-property-inheritance` setting.

In this example, '`:cache`' defaults to '`yes`' for all code blocks in the sub-tree.

```
* sample header
  :PROPERTIES:
  :header-args:    :cache yes
  :END:
```

Properties defined through `org-set-property` function, bound to *C-c C-x p*, apply to all active languages. They override properties set in `org-babel-default-header-args`.

Language-specific header arguments are also read from properties '`header-args:<LANG>`' where <*LANG*> is the language identifier. For example, 例えば

```
* Heading
  :PROPERTIES:
  :header-args:clojure:    :session *clojure-1*
  :header-args:R:          :session *R*
  :END:
** Subheading
  :PROPERTIES:
  :header-args:clojure:    :session *clojure-2*
  :END:
```

would force 強制します separate sessions for Clojure blocks in '`Heading`' and '`Subheading`', but use the same session for all R blocks. Blocks in '`Subheading`' inherit settings from '`Heading`'.

Code block specific header arguments

Header arguments are most commonly set at the source code block level, on the '`#+BEGIN_SRC`' line. Arguments set at this level take precedence over those set in the `org-babel-default-header-args` variable, and also those set as header properties.

In the following example, setting '`:results`' to '`silent`' makes it ignore results of the code execution. Setting '`:exports`' to '`code`' exports エクスポートします only the body 本体だけを of the code block to HTML or LaTeX.

```
#+NAME: factorial
#+BEGIN_SRC haskell :results silent :exports code :var n=0
  fac 0 = 1
  fac n = n * fac (n-1)
#+END_SRC
```

The same header arguments in an inline code block:

```
src_haskell[:exports both]{fac 5}
```

Code block header arguments can span multiple lines using '`#+HEADER:`' on each line. Note 注意してください that Org currently accepts the plural spelling of '`#+HEADER:`' only as a convenience for backward-compatibility. It may be removed at some point.

Multi-line header arguments on an unnamed code block:

```
#+HEADER: :var data1=1
#+BEGIN_SRC emacs-lisp :var data2=2
   (message "data1:%S, data2:%S" data1 data2)
#+END_SRC

#+RESULTS:
: data1:1, data2:2
```

Multi-line header arguments on a named code block:

```
#+NAME: named-block
#+HEADER: :var data=2
#+BEGIN_SRC emacs-lisp
  (message "data:%S" data)
#+END_SRC

#+RESULTS: named-block
  : data:2
```

Header arguments in function calls

Header arguments in function calls are the most specific and override all other settings in case of an overlap. They get the highest priority. Two '`#+CALL:`' examples are shown below. For the complete syntax of '`CALL`' keyword, see 節 14.4 [Evaluating Code Blocks], ページ 221, を見てください

In this example, '`:exports results`' header argument is applied to the evaluation of the '`#+CALL:`' line.

```
#+CALL: factorial(n=5) :exports results
```

In this example, '`:session special`' header argument is applied to the evaluation of '`factorial`' code block.

```
#+CALL: factorial[:session special](n=5)
```

14.3 Environment of a Code Block

Passing arguments

Use 使ってください'`var`' for passing arguments to source code blocks. The specifics of variables in code blocks vary by the source language and are covered in the language-specific documentation. The syntax for '`var`', however, しかし is the same 同じです for all languages. This includes declaring a variable, and assigning a default value.

The following syntax is used to pass arguments to code blocks using the '`var`' header argument.

```
:var NAME=ASSIGN
```

NAME is the name of the variable bound in the code block body. *ASSIGN* is a literal value, such as a string, 文字列、a number, a reference to a table, a list, a literal example, another code block---with or without arguments---or the results of evaluating a code block.

Here これらは are examples of passing values by reference:

table A table named with a '`NAME`' keyword.

```
#+NAME: example-table
| 1 |
| 2 |
| 3 |
| 4 |

#+NAME: table-length
#+BEGIN_SRC emacs-lisp :var table=example-table
  (length table)
#+END_SRC

#+RESULTS: table-length
: 4
```

When passing a table, you can treat specially the row, or the column, containing 含んでいる labels for the columns, or the rows, in the table.

The '`colnames`' header argument accepts '`yes`', '`no`', or '`nil`' values. The default value デフォルト値は is '`nil`' です: if an input table has column names---because the second row is a horizontal rule---then Org removes the column names, processes the table, puts back the column names, and then writes the table to the results block. Using '`yes`', Org does the same to the first row, even if the initial table does not contain any horizontal rule. When set 設定されているときには、to '`no`' に Org does not pre-process column names at all.

```
#+NAME: less-cols
| a |
|---|
| b |
| c |

#+BEGIN_SRC python :var tab=less-cols :colnames nil
  return [[val + '*' for val in row] for row in tab]
#+END_SRC

#+RESULTS:
| a  |
|----|
| b* |
| c* |
```

Similarly, the '`rownames`' header argument can take two values: '`yes`' or '`no`'. When set to '`yes`' に設定されているときには、Org removes the first column, processes the table, puts back the first column, and then writes the table to the

results block. The default is デフォルト値は‘no’ であり、which means 意味します Org does not pre-process the first column. Note 注意してください that Emacs Lisp code blocks ignore ‘rownames’ header argument because of the ease of table-handling in Emacs.

```
#+NAME: with-rownames
| one | 1 | 2 | 3 | 4 |  5 |
| two | 6 | 7 | 8 | 9 | 10 |

#+BEGIN_SRC python :var tab=with-rownames :rownames yes
  return [[val + 10 for val in row] for row in tab]
#+END_SRC

#+RESULTS:
| one | 11 | 12 | 13 | 14 | 15 |
| two | 16 | 17 | 18 | 19 | 20 |
```

list A simple named list.

```
#+NAME: example-list
- simple
  - not
  - nested
- list

#+BEGIN_SRC emacs-lisp :var x=example-list
  (print x)
#+END_SRC

#+RESULTS:
| simple | list |
```

Note 注意してください that only the top level list items are passed along. Nested list items are ignored.

code block without arguments

A code block name, as assigned by ‘NAME’ keyword from the example above, optionally followed by parentheses.

```
#+BEGIN_SRC emacs-lisp :var length=table-length()
  (* 2 length)
#+END_SRC

#+RESULTS:
: 8
```

code block with arguments

A code block name, as assigned by ‘NAME’ keyword, followed by parentheses and optional arguments passed within the parentheses.

```
#+NAME: double
#+BEGIN_SRC emacs-lisp :var input=8
  (* 2 input)
```

```
#+END_SRC

#+RESULTS: double
: 16

#+NAME: squared
#+BEGIN_SRC emacs-lisp :var input=double(input=1)
  (* input input)
#+END_SRC

#+RESULTS: squared
: 4
```

literal example

A literal example block named with a '`NAME`' keyword.

```
#+NAME: literal-example
#+BEGIN_EXAMPLE
  A literal example
  on two lines
#+END_EXAMPLE

#+NAME: read-literal-example
#+BEGIN_SRC emacs-lisp :var x=literal-example
  (concatenate #'string x " for you.")
#+END_SRC

#+RESULTS: read-literal-example
: A literal example
: on two lines for you.
```

Indexing variable values enables referencing portions of a variable. Indexes are 0 based with negative values counting backwards from the end. If an index is separated by commas then each subsequent section indexes as the next dimension. Note 注意してください that this indexing occurs *before* other table-related header arguments are applied, such as '`hlines`', '`colnames`' and '`rownames`'. The following example assigns the last cell of the first row the table '`example-table`' to the variable '`data`':

```
#+NAME: example-table
| 1 | a |
| 2 | b |
| 3 | c |
| 4 | d |

#+BEGIN_SRC emacs-lisp :var data=example-table[0,-1]
  data
#+END_SRC

#+RESULTS:
: a
```

Two integers separated by a colon reference a range of variable values. In that case その場合には the entire inclusive range is referenced. For example 例えば the following assigns the middle three rows of ‘example-table’ to ‘data’.

```
#+NAME: example-table
| 1 | a |
| 2 | b |
| 3 | c |
| 4 | d |
| 5 | 3 |

#+BEGIN_SRC emacs-lisp :var data=example-table[1:3]
  data
#+END_SRC

#+RESULTS:
| 2 | b |
| 3 | c |
| 4 | d |
```

To pick the entire range, use an empty index, or the single character ‘*’. =0:-1 = does the same thing. Example below shows 示しています how to reference the first column only.

```
#+NAME: example-table
| 1 | a |
| 2 | b |
| 3 | c |
| 4 | d |

#+BEGIN_SRC emacs-lisp :var data=example-table[,0]
  data
#+END_SRC

#+RESULTS:
| 1 | 2 | 3 | 4 |
```

Index referencing インデックス参照を can be used 使うことができます for tables 表と and code blocks. コードブロックに対して Index referencing インデックス参照は can handle any number of dimensions. Commas delimit multiple dimensions, as shown below.

```
#+NAME: 3D
#+BEGIN_SRC emacs-lisp
  '(((1  2  3)  (4  5  6)  (7  8  9))
    ((10 11 12) (13 14 15) (16 17 18))
    ((19 20 21) (22 23 24) (25 26 27)))
#+END_SRC

#+BEGIN_SRC emacs-lisp :var data=3D[1,,1]
  data
#+END_SRC
```

```
#+RESULTS:
| 11 | 14 | 17 |
```

Note 注意してください that row names and column names are not removed prior to variable indexing. You need to take them into account, それらを考慮する必用がありま even when ‘`colnames`’ or ‘`rownames`’ header arguments remove them.

Emacs lisp code can also set the values for variables. To differentiate a value from Lisp code, Org interprets any value starting with ‘(’, ‘[’, ‘'’ or ‘`’ as Emacs Lisp code. The result of evaluating that code is then assigned to the value of that variable. The following example shows 示しています how to reliably query and pass the file name of the Org-mode buffer to a code block using headers. We need reliability here because the file's name could change once the code in the block starts executing.

```
#+BEGIN_SRC sh :var filename=(buffer-file-name) :exports both
  wc -w $filename
#+END_SRC
```

Note 注意してください that values read from tables and lists are not mistakenly evaluated as Emacs Lisp code, as illustrated in the following example.

```
#+NAME: table
| (a b c) |

#+HEADER: :var data=table[0,0]
#+BEGIN_SRC perl
  $data
#+END_SRC

#+RESULTS:
: (a b c)
```

Using sessions

Two code blocks can share the same environment. The ‘`session`’ header argument is for running multiple source code blocks under one session. Org runs code blocks with the same session name in the same interpreter process.

‘`none`’ デフォルトです。Each code block gets a new interpreter process to execute. The process terminates once the block is evaluated.

STRING Any string besides ‘`none`’ turns that string into the name of that session. For example, 例えば‘`:session STRING`’ names it ‘`STRING`’. If ‘`session`’ has no value, then the session name is derived from the source language identifier. Subsequent blocks with the same source code language use the same session. Depending on 依存して the language, 言語に state variables, code from other blocks, and the overall interpreted environment may be shared. Some interpreted languages support concurrent sessions when subsequent source code language blocks change session names.

Only languages that provide interactive evaluation can have session support. Not all languages provide this support, such as C や ditaa といった Even languages, such as Python や Haskell といった, that do support interactive evaluation impose limitations on

allowable language constructs that can run interactively. インタラクティブに Org inherits those limitations for those code blocks running in a session.

Choosing a working directory

The '`dir`' ヘッダー引数は specifies 指定します the default directory during code block execution. If it is absent, then the directory associated with the current buffer is used. In other words, supplying '`:dir PATH`' temporarily has the same effect as changing the current directory with *`M-x cd PATH`*, and then not setting '`dir`'. Under the surface, '`dir`' simply 単純に sets 設定します the value 値を of the Emacs variable `default-directory`.

For example, 例えば to save the plot file in the '`Work/`' folder of the home directory---notice tilde is expanded:

```
#+BEGIN_SRC R :file myplot.png :dir ~/Work
  matplot(matrix(rnorm(100), 10), type="l")
#+END_SRC
```

To evaluate the code block on a remote machine, supply a remote directory name using Tramp syntax. 例です:

```
#+BEGIN_SRC R :file plot.png :dir /scp:dand@yakuba.princeton.edu:
  plot(1:10, main=system("hostname", intern=TRUE))
#+END_SRC
```

Org first 最初に captures the text results as usual for insertion in the Org file. Then Org also inserts 挿入します a link リンクを to the remote file, thanks to Emacs Tramp. Org constructs the remote path to the file name from '`dir`' and `default-directory`, as illustrated here:

```
[[file:/scp:dand@yakuba.princeton.edu:/home/dand/plot.png][plot.png]]
```

When '`dir`' is used with '`session`', Org sets the starting directory for a new session. But しかし Org does not alter the directory of an already existing session.

Do not use '`dir`' with '`:exports results`' or with '`:exports both`' to avoid Org inserting incorrect links to remote files. That is because Org does not expand `default directory` to avoid some underlying portability issues.

Inserting headers and footers

The '`prologue`' header argument is for appending to the top of the code block for execution, like a reset instruction. For example, 例えば you may use '`:prologue "reset"`' を in a Gnuplot code block or, for every such block:

```
(add-to-list 'org-babel-default-header-args:gnuplot
             '((:prologue . "reset")))
```

Likewise, the value of the '`epilogue`' header argument is for appending to the end of the code block for execution.

14.4 Evaluating Code Blocks

A note about security: With code evaluation comes the risk of harm. Org safeguards by prompting for user's permission before executing any code in the source block. To customize

カスタマイズしてください this safeguard, or disable it, see 節 15.5 [Code Evaluation Security], ページ 242.

How to evaluate source code

Org captures the results of the code block evaluation and inserts 挿入します them in the Org file, right after the code block. The insertion point is after a newline and the '`RESULTS`' keyword. Org creates 作成します the '`RESULTS`' keyword if one is not already there.

By default, デフォルトで Org enables only Emacs Lisp code blocks for execution. See 節 14.8 [Languages], ページ 232, to enable other languages.

Org provides 提供しています many ways 多くの方法を to execute code blocks. コードブロックを実行するための`C-c C-c` or `C-c C-v e` with the point on a code block[1] calls the `org-babel-execute-src-block` function, which これは executes 実行します the code in the block, collects the results, and inserts 挿入します them in the buffer.

By calling 呼び出すことによって a named code block 名前つきのコードブロックを

Limit code block evaluation

The '`eval`' header argument can limit evaluation of specific code blocks and '`CALL`' keyword. It is useful 役に立ちます for protection against evaluating untrusted code blocks by prompting for a confirmation.

'`never`' or '`no`'
: Org never evaluates the source code.

'`query`'
: Org prompts プロンプトを出します the user for permission to evaluate the source code.

'`never-export`' or '`no-export`'
: Org does not evaluate the source code ソースコードを when exporting, エクスポートするときに yet the user can evaluate it それを interactively. インタラクティブに

'`query-export`'
: Org prompts プロンプトを出します the user for permission to evaluate the source code during export.

If '`eval`' header argument is not set, then Org determines whether to evaluate the source code from the `org-confirm-babel-evaluate` variable（節 15.5 [Code Evaluation Security], ページ 242, を見てください）.

Cache results of evaluation

The '`cache`' header argument is for caching results of evaluating code blocks. Caching results can avoid re-evaluating a code block that have not changed since the previous run. To benefit from the cache and avoid redundant evaluations, the source block must have a result already present in the buffer, and neither the header arguments---including 含む the value 値を of '`var`' references ---nor the text of the block itself has changed since the result

[1] The オプション`org-babel-no-eval-on-ctrl-c-ctrl-c` を can be used 使うことができます。to remove code evaluation コードの実行を取り除くために、from the `C-c C-c` というキーバインディングから

was last computed. This feature greatly helps avoid long-running calculations. For some edge cases, however, しかし the cached results may not be reliable.

The caching feature is best for when code blocks are pure functions, that is functions that return the same value for the same input arguments（節 14.3 [Environment of a Code Block], ページ 215, を見てください）and that do not have side effects, and do not rely on external variables other than the input arguments. Functions that depend on a timer, file system objects, and random number generators are clearly unsuitable for caching.

A note of warning: when '`cache`' is used in a session, caching may cause unexpected results.

When the caching mechanism tests for any source code changes, it does not expand Noweb style references（節 14.10 [Noweb Reference Syntax], ページ 234, を見てください）。For reasons why, see `http://thread.gmane.org/gmane.emacs.orgmode/79046` を見てください。

The '`cache`' header argument can have one of two values: '`yes`' or '`no`'.

'`no`' デフォルトです。No caching of results; code block evaluated every time.

'`yes`' Whether to run the code or return the cached results is determined by comparing the SHA1 hash value of the combined code block and arguments passed to it. This hash value is packed on the '`#+RESULTS:`' line from previous evaluation. When hash values match, Org does not evaluate the code block. When hash values mismatch, Org evaluates the code block, inserts 挿入します the results, 結果を recalculates the hash value, and updates '`#+RESULTS:`' line.

In this example, both functions are cached. But 'しかし `caller`' runs only if the result from '`random`' has changed since the last run.

```
#+NAME: random
#+BEGIN_SRC R :cache yes
  runif(1)
#+END_SRC

#+RESULTS[a2a72cd647ad44515fab62e144796432793d68e1]: random
0.4659510825295

#+NAME: caller
#+BEGIN_SRC emacs-lisp :var x=random :cache yes
  x
#+END_SRC

#+RESULTS[bec9c8724e397d5df3b696502df3ed7892fc4f5f]: caller
0.254227238707244
```

14.5 Results of Evaluation

How Org handles results of a code block execution depends on many header arguments working together. The primary determinant, 主な決定要素は however, しかし is the '`results`' header argument. It accepts four classes of options. Each code block can take only one option per class:

collection For how the results should be collected from the code block;

type For which type of result the code block will return; affects how Org processes and inserts 挿入します results 結果を in the Org buffer;

format For the result; affects how Org processes and inserts 挿入します results 結果を in the Org buffer;

handling For processing results after evaluation of the code block;

Collection

Collection options specify the results. Choose one of the options; they are mutually exclusive.

‘`value`’ デフォルトです。Functional mode. Org gets 得ま the value by wrapping the code in a function definition in the language of the source block. That それが is why when using 使うときには‘`:results value`’ を code should execute like a function and return a value. For languages like Python, an explicit `return` statement is mandatory when using 使うときには‘`:results value`’ を Result is the value returned by the last statement in the code block.

When evaluating the code block in a session（節 14.3 [Environment of a Code Block], ページ 215, を見てください）Org passes the code to an interpreter running as an interactive Emacs inferior process. Org gets the value from the source code interpreter’s last statement output. Org has to use language-specific methods to obtain the value. For example, 例えば from the variable _ in Python and Ruby, and the value of `.Last.value` in R.

‘`output`’ Scripting mode. Org passes the code to an external process running the interpreter. Org returns the contents of the standard output stream as text results.

When using 使うときには a session, セッションを Org passes the code to the interpreter running as an interactive Emacs inferior process. Org concatenates any text output from the interpreter and returns the collection as a result.

Note 注意してください that this collection is not the same as that would be collected from stdout of a non-interactive interpreter running as an external process. Compare for example 例えば these two blocks:

```
#+BEGIN_SRC python :results output
  print "hello"
  2
  print "bye"
#+END_SRC

#+RESULTS:
: hello
: bye
```

In the above non-session mode, the "2" is not printed; so it does not appear in results.

```
#+BEGIN_SRC python :results output :session
  print "hello"
```

```
  2
  print "bye"
#+END_SRC

#+RESULTS:
: hello
: 2
: bye
```

In the above session, the interactive interpreter receives and prints `"2"`. Results show that.

Type

Type tells what result types to expect from the execution of the code block. Choose one of the options; they are mutually exclusive. The default behavior is to automatically 自動的に determine 決定します the result type. 結果の種類を

‘`table`’
‘`vector`’ Interpret the results as an Org table. If the result is a single value, create a table with one row and one column. Usage example: ‘`:results value table`’.

In-between each table row or below the table headings, sometimes ときどき results have horizontal lines, which これらは are also known 知られています as `"hlines"`. The ‘`hlines`’ argument with the default ‘`no`’ value strips such lines from the input table. For most code, this is desirable, or else those ‘`hline`’ symbols raise unbound variable errors. A ‘`yes`’ accepts such lines, as demonstrated in the following example.

```
#+NAME: many-cols
| a | b | c |
|---+---+---|
| d | e | f |
|---+---+---|
| g | h | i |

#+NAME: no-hline
#+BEGIN_SRC python :var tab=many-cols :hlines no
  return tab
#+END_SRC

#+RESULTS: no-hline
| a | b | c |
| d | e | f |
| g | h | i |

#+NAME: hlines
#+BEGIN_SRC python :var tab=many-cols :hlines yes
  return tab
#+END_SRC
```

```
#+RESULTS: hlines
| a | b | c |
|---+---+---|
| d | e | f |
|---+---+---|
| g | h | i |
```

‘`list`’ Interpret the results as an Org list. If the result is a single value, create a list of one element.

‘`scalar`’
‘`verbatim`’
Interpret literally and insert as quoted text. Do not create a table. Usage example: ‘`:results value verbatim`’.

‘`file`’ Interpret as a filename. Save the results of execution of the code block to that file, then insert a link to it. You can control both the filename and the description associated to the link.

Org first tries to generate the filename from the value of the ‘`file`’ header argument and the directory specified using the ‘`output-dir`’ header arguments. If ‘`output-dir`’ が is not specified, Org assumes it それが is the current directory.

```
#+BEGIN_SRC asymptote :results value file :file circle.pdf :output-dir img/
  size(2cm);
  draw(unitcircle);
#+END_SRC
```

If ‘`file`’ is missing, Org generates 生成します the base name of the output file from the name of the code block, and its extension from the ‘`file-ext`’ header argument. In that case, その場合には both the name and the extension are mandatory 必須です[2].

```
#+name: circle
#+BEGIN_SRC asymptote :results value file :file-ext pdf
  size(2cm);
  draw(unitcircle);
#+END_SRC
```

The ‘`file-desc`’ header argument defines the description（節 4.1 [Link Format], ページ 40, を見てください）for the link. If ‘`file-desc`’ has no value, Org uses 使います the generated file name for both the "link" and "description" parts of the link.

By default, デフォルトで Org assumes 仮定します that a table written to a file has TAB-delimited output. You can choose a different separator with the ‘`sep`’ header argument.

[2] Due to the way this header argument is implemented, it implies ":results file". Therefore if it is set for multiple blocks at once (by a subtree or buffer property for example), 例えば all blocks are forced to produce file results. This is seldom desired behavior, so it is recommended to set this header only on a per-block basis. It is possible that this aspect of the implementation might change in the future.

Format

Format pertains to the type of the result returned by the code block. Choose one of the options; they are mutually exclusive. The default デフォルトは follows from the type specified above.

‘`code`’ Result enclosed in a code block. Useful 役に立ちます for parsing. Usage example: ‘`:results value code`’.

‘`drawer`’ Result wrapped in a ‘`RESULTS`’ drawer. Useful 役に立ちます for containing 含んでいる‘`raw`’ or ‘`org`’ results for later scripting and automated processing. Usage example: ‘`:results value drawer`’.

‘`html`’ Results enclosed in a ‘`BEGIN_EXPORT html`’ block. Usage example: ‘`:results value html`’.

‘`latex`’ Results enclosed in a ‘`BEGIN_EXPORT latex`’ block. Usage example: ‘`:results value latex`’.

‘`link`’
‘`graphics`’
Result is a link to the file specified in ‘`:file`’ header argument. However, しかし unlike plain ‘`:file`’, nothing is written to the disk. The block is used for its side-effects only, as in the following example:

```
#+begin_src shell :results link :file "download.tar.gz"
wget -c "http://example.com/download.tar.gz"
#+end_src
```

‘`org`’ Results enclosed in a ‘`BEGIN_SRC org`’ block. For comma-escape, either `TAB` in the block, or export the file. Usage example: ‘`:results value org`’.

‘`pp`’ Result converted to pretty-print source code. Enclosed in a code block. Languages supported: Emacs Lisp, Python, and Ruby. Usage example: ‘`:results value pp`’.

‘`raw`’ Interpreted as raw Org-mode. Inserted directly 直接 into the buffer. Aligned if it is a table. Usage example: ‘`:results value raw`’.

The ‘`wrap`’ header argument unconditionnally marks the results block by appending strings to ‘`#+BEGIN_`’ and ‘`#+END_`’. If no string is specified, Org wraps the results in a ‘`#+BEGIN_results`’ ... ‘`#+END_results`’ block. It takes precedent over the ‘`results`’ value listed above. E.g., 例えば

```
#+BEGIN_SRC emacs-lisp :results html :wrap EXPORT markdown
"<blink>Welcome back to the 90's</blink>"
#+END_SRC

#+RESULTS:
#+BEGIN_EXPORT markdown
<blink>Welcome back to the 90's</blink>
#+END_EXPORT
```

Handling

Handling options after collecting the results.

‘`silent`’ Do not insert results in the Org-mode buffer, but echo them in the minibuffer. Usage example: ‘`:results output silent`’.

‘`replace`’ デフォルトです。Insert results in the Org buffer. Remove previous results. Usage example: ‘`:results output replace`’.

‘`append`’ Append results to the Org buffer. Latest results are at the bottom. Does not remove previous results. Usage example: ‘`:results output append`’.

‘`prepend`’ Prepend results to the Org buffer. Latest results are at the top. Does not remove previous results. Usage example: ‘`:results output prepend`’.

Post-processing

The ‘`post`’ header argument is for post-processing results from block evaluation. When ‘`post`’ に has any value, Org binds the results to `*this*` variable for easy passing to ‘`var`’ header argument specifications（節 14.3 [Environment of a Code Block], ページ 215, を見てください）。That makes results available to other code blocks, or even for direct Emacs Lisp code execution.

The following two examples illustrate ‘`post`’ header argument in action. The first one shows 示しています how to attach an ‘`ATTR_LATEX`’ keyword using ‘`post`’.

```
#+NAME: attr_wrap
#+BEGIN_SRC sh :var data="" :var width="\\textwidth" :results output
  echo "#+ATTR_LATEX: :width $width"
  echo "$data"
#+END_SRC

#+HEADER: :file /tmp/it.png
#+BEGIN_SRC dot :post attr_wrap(width="5cm", data=*this*) :results drawer
  digraph{
          a -> b;
          b -> c;
          c -> a;
  }
#+end_src

#+RESULTS:
:RESULTS:
#+ATTR_LATEX :width 5cm
[[file:/tmp/it.png]]
:END:
```

The second example shows 示しています use 使用方法を of ‘`colnames`’ header argument in ‘`post`’ to pass data between code blocks.

```
#+NAME: round-tbl
#+BEGIN_SRC emacs-lisp :var tbl="" fmt="%.3f"
  (mapcar (lambda (row)
```

```
              (mapcar (lambda (cell)
                        (if (numberp cell)
                            (format fmt cell)
                          cell))
                      row))
            tbl)
#+end_src

#+BEGIN_SRC R :colnames yes :post round-tbl[:colnames yes](*this*)
  set.seed(42)
  data.frame(foo=rnorm(1))
#+END_SRC

#+RESULTS:
|   foo |
|-------|
| 1.371 |
```

14.6 Exporting Code Blocks

It is possible 可能です to export the *code* of code blocks, the *results* of code block evaluation, *both* the code and the results of code block evaluation, or *none*. Org defaults to exporting *code* for most languages. For some languages, such as ditaa といった Org Org-mode は defaults to *results*. To export エクスポートするには just the body 本体だけを of code blocks, コードブロックの see 節 11.6 [Literal Examples], ページ 141, を見てください。 To selectively 選択的に export エクスポートするには sub-trees サブツリーを of an Org document, Org-mode のドキュメントの see 章 12 [Exporting], ページ 145, を見てください。

The '`exports`' header argument is to specify if that part of the Org file is exported to, say, HTML or LaTeX formats.

'`code`' The default. デフォルトです。The body of code is included into the exported file. Example: '`:exports code`'.

'`results`' The results of evaluation of the code is included in the exported file. Example: '`:exports results`'.

'`both`' Both the code and results of evaluation are included in the exported file. Example: '`:exports both`'.

'`none`' Neither the code nor the results of evaluation is included in the exported file. Whether the code is evaluated at all depends on other options. 例: '`:exports none`'.

To stop 止めるには Org Org-mode が from evaluating 評価すうことを code blocks コードブロックを to speed exports, use the header argument '`:eval never-export`'（節 14.4 [Evaluating Code Blocks], ページ 221, を見てください）。To stop 止めるには Org Org-mode が from evaluating 評価することを code blocks コードブロックを for greater security, set 設定してださい the `org-export-use-babel` 変数を to `nil` に but understand that header arguments will have no effect.

Turning off evaluation comes in handy 便利です when batch processing. For example, 例えば markup languages for wikis, which これは have あります a high risk 高いリスクが of untrusted code. Stopping code block evaluation also stops evaluation of all header arguments of the code block. This これは may not be desirable in some circumstances. いくつかの状況では So during export, to allow evaluation of just the header arguments ヘッダー引数だけ but not any code evaluation in the source block, set '`:eval never-export`'（節 14.4 [Evaluating Code Blocks], ページ 221, を見てください）。

Org never evaluates code blocks in commented sub-trees when exporting エクスポートするときに（節 12.6 [Comment Lines], ページ 153, を見てください）。On the other hand, 反対に Org Org-mode は does evaluate code blocks コードブロックを in sub-trees excluded from export（節 12.2 [Export Settings], ページ 146, を見てください）。

14.7 Extracting Source Code

Extracting source code from code blocks is a basic task in literate programming. Org has features to make this easy. In literate programming parlance, documents on creation are *woven* with code and documentation, and on export, the code is tangled for execution by a computer. Org facilitates weaving and tangling for producing, maintaining, sharing, and exporting literate programming documents. Org provides 提供しています extensive customization options for extracting source code.

When Org tangles code blocks, it expands, merges, and transforms them. Then Org recomposes them into one or more separate files, as configured through the options. During this tangling process, Org expands variables in the source code, and resolves any Noweb style references（節 14.10 [Noweb Reference Syntax], ページ 234, を見てください）。

Header arguments

The '`tangle`' ヘッダー引数は specifies 指定します if the code block コードブロックが is exported エクスポートされるかどうかを to source file(s).

'`yes`' Export エクスポートします the code block to source file. The file name for the source file is derived from the name of the Org file, and the file extension is derived from the source code language identifier. Example: '`:tangle yes`'.

'`no`' The default. デフォルトです。Do not extract the code in a source code file. Example: '`:tangle no`'.

FILENAME
Export エクスポートします the code block to source file whose file name is derived from any string passed to the '`tangle`' header argument. Org derives the file name as being relative to the directory of the Org file's location. Example: '`:tangle FILENAME`'.

The '`mkdirp`' header argument creates 作成します parent directories for tangled files if the directory does not exist. '`yes`' enables directory creation and '`no`' inhibits directory creation.

The '`comments`' header argument controls inserting comments into tangled files. These are above and beyond whatever comments may already exist in the code block.

'`no`' The default. デフォルトです。Do not insert any extra comments during tangling.

‘`link`’ Wrap the code block in comments. Include links pointing back to the place in the Org file from where the code was tangled.

‘`yes`’ Kept for backward compatibility; same as ‘`link`’.

‘`org`’ Nearest headline text from Org file is inserted as comment. The exact text that is inserted is picked from the leading context of the source block.

‘`both`’ Includes both ‘`link`’ and ‘`org`’ options.

‘`noweb`’ Includes ‘`link`’ option, expands Noweb references（節 14.10 [Noweb Reference Syntax], ページ 234, を見てください）、and wraps them in link comments inside the body of the code block.

The ‘`padline`’ header argument controls insertion of newlines to pad source code in the tangled file.

‘`yes`’ デフォルトです。Insert a newline before and after each code block in the tangled file.

‘`no`’ Do not insert newlines to pad the tangled code blocks.

The ‘`shebang`’ header argument can turn results into executable script files. By setting 設定することによって it それを to a string value ---for example, 例えば‘`:shebang "#!/bin/bash"`’ ---Org inserts 挿入します that string その文字列を as the first line of the tangled file that the code block is extracted to. Org then turns on the tangled file’s executable permission.

The ‘`tangle-mode`’ header argument specifies 指定します what permissions to set for tangled files by `set-file-modes`. For example, 例えば to make a read-only tangled file, use ‘`:tangle-mode (identity #o444)`’. To make it executable, use ‘`:tangle-mode (identity #o755)`’. It also overrides executable permission granted by ‘`shebang`’. When multiple source code blocks tangle to a single file with different and conflicting ‘`tangle-mode`’ header arguments, Org’s behavior is undefined.

By default デフォルトで Org expands 展開します code blocks during tangling. The ‘`no-expand`’ header argument turns off such expansions. Note 注意してください that one side-effect of expansion by `org-babel-expand-src-block` also assigns values（節 14.3 [Environment of a Code Block], ページ 215, を見てください）to variables. Expansions also replace Noweb references with their targets（節 14.10 [Noweb Reference Syntax], ページ 234, を見てください）。Some of these expansions may cause premature assignment, hence よって this option. This option makes a difference only for tangling. It has no effect when exporting エクスポートするときに since code blocks for execution have to be expanded anyway.

Functions

`org-babel-tangle`

Tangle the current file. Bound to `C-c C-v t`.

With prefix argument only tangle the current code block.

`org-babel-tangle-file`

Choose a file to tangle. Bound to `C-c C-v f`.

Hooks

`org-babel-post-tangle-hook`
: This hook is run from within code files tangled by `org-babel-tangle`, making it suitable for post-processing, compilation, and evaluation of code in the tangled files.

Jumping between code and Org

Debuggers デバッガは normally 通常は link errors and messages back to the source code. But しかし for tangled files, we want to link back to the Org file, not to the tangled source file. To make this extra jump, Org uses `org-babel-tangle-jump-to-org` function with two additional source code block header arguments:

1. Set '`padline`' to true---this is the default setting.
2. Set '`comments`' to '`link`', which これは

   makes Org insert 挿入させます links to the Org file.

14.8 Languages

Code blocks in the following languages are supported.

Language	Identifier	Language	Identifier
Asymptote	'`asymptote`'	Lisp	'`lisp`'
Awk	'`awk`'	Lua	'`lua`'
C	'`C`'	MATLAB	'`matlab`'
C++	'`C++`'[3]	Mscgen	'`mscgen`'
Clojure	'`clojure`'	Objective Caml	'`ocaml`'
CSS	'`css`'	Octave	'`octave`'
D	'`D`'[4]	Org-mode	'`org`'
ditaa	'`ditaa`'	Oz	'`oz`'
Emacs Calc	'`calc`'	Perl	'`perl`'
Emacs Lisp	'`emacs-lisp`'	Plantuml	'`plantuml`'
Eshell	'`eshell`'	Processing.js	'`processing`'
Fortran	'`fortran`'	Python	'`python`'
Gnuplot	'`gnuplot`'	R	'`R`'
GNU Screen	'`screen`'	Ruby	'`ruby`'
Graphviz	'`dot`'	Sass	'`sass`'
Haskell	'`haskell`'	Scheme	'`scheme`'
Java	'`java`'	Sed	'`sed`'
Javascript	'`js`'	shell	'`sh`'
LaTeX	'`latex`'	SQL	'`sql`'
Ledger	'`ledger`'	SQLite	'`sqlite`'
Lilypond	'`lilypond`'	Vala	'`vala`'

[3] `C++` language is handled in '`ob-C.el`'. Even though the identifier for such source blocks is '`C++`', you activate it by loading the C language.

[4] D language is handled in '`ob-C.el`'. Even though the identifier for such source blocks is '`D`', you activate it by loading the C language.

Additional documentation for some languages is at `https://orgmode.org/worg/org-contrib/babel/languages.html`.

By default, デフォルトで only Emacs Lisp is enabled for evaluation. To enable 有効にするか or disable other languages, customize カスタマイズしてください the `org-babel-load-languages` 変数を either through the Emacs customization interface, or by adding code to the init file as shown next.

In this example, evaluation is disabled for Emacs Lisp, and enabled for R.

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

Note 注意してください that this これが is not the only way to enable a language. Org also enables languages when loaded with `require` statement. For example, 例えば the following enables execution of Clojure code blocks:

```
(require 'ob-clojure)
```

14.9 Editing Source Code

Use 使ってください*C-c '* を to edit the current code block. It opens a new major-mode edit buffer containing 含んでいる the body 本体を of the source code block, ready 準備ができている for any edits. 編集の Use 使ってください*C-c '* を again もう一度 to close the buffer and return to the Org buffer.

C-x C-s saves the buffer and updates the contents of the Org buffer. Set `org-edit-src-auto-save-idle-delay` to save the base buffer after a certain idle delay time. Set `org-edit-src-turn-on-auto-save` to auto-save this buffer into a separate file using Auto-save mode.

While editing the source code in the major mode, the Org Src minor mode remains active. It それは provides 提供しています these customization variables as described below. 下で説明されている For even more variables, look in the customization group `org-edit-structure`.

`org-src-lang-modes`
: If an Emacs major-mode named `<LANG>-mode` exists, where *<LANG>* is the language identifier from code block's header line, then the edit buffer uses that major mode. Use 使ってください this variable to arbitrarily map language identifiers to major modes.

`org-src-window-setup`
: For specifying Emacs window arrangement when the new edit buffer is created. 作成されます

`org-src-preserve-indentation`
: デフォルトは`nil` です。Source code is indented. This indentation applies 適用されます during export or tangling, and depending on 依存して the context, コンテキストに may alter leading spaces and tabs. When non-`nil`, `nil` でない値のときには source code ソースコードは is aligned with the leftmost column. No lines are modified during export or tangling, which これが is very useful とても役に立ちます for white-space sensitive languages, such as Python といった

`org-src-ask-before-returning-to-edit-buffer`
: When `nil` のときには、Org Org-mode は returns to the edit buffer without further prompts. さらにプロンプトを出すことなく The default デフォルトは prompts プロンプトを出します for a confirmation. 確認を求めて

Set 設定してください`org-src-fontify-natively` を`nil` でない値に to turn on オンにするにっは native code fontification in the *Org* buffer. Fontification of code blocks can give 与えてくれることができます visual separation 視覚的な区別を of text and code on the display page. To further customize さらにカスタマイズするには the appearance 見かけを of `org-block` for specific languages, customize カスタマイズしてください`org-src-block-faces` を The following example shades the background of regular blocks, and colors source blocks only for Python and Emacs Lisp languages.

```
(require 'color)
(set-face-attribute 'org-block nil :background
                    (color-darken-name
                     (face-attribute 'default :background) 3))

(setq org-src-block-faces '(("emacs-lisp" (:background "#EEE2FF"))
                            ("python" (:background "#E5FFB8"))))
```

14.10 Noweb Reference Syntax

Org Org-mode は supports サポートしています named blocks in Noweb[5] style syntax:

```
<<CODE-BLOCK-ID>>
```

Org can replace the construct with the source code, or the results of evaluation, of the code block identified as *CODE-BLOCK-ID*.

The '`noweb`' header argument controls expansion of Noweb syntax references. Expansions occur 起きます when source code blocks are evaluated, tangled, or exported.

'`no`'
: デフォルトです。No expansion of Noweb syntax references in the body of the code when evaluating, tangling, or exporting.

'`yes`'
: Expansion of Noweb syntax references in the body of the code block when evaluating, tangling, or exporting.

'`tangle`'
: Expansion of Noweb syntax references in the body of the code block when tangling. No expansion when evaluating or exporting.

'`no-export`'
: Expansion of Noweb syntax references in the body of the code block when evaluating or tangling. No expansion when exporting. エクスポートするときに

'`strip-export`'
: Expansion of Noweb syntax references in the body of the code block when expanding prior to evaluating or tangling. Removes 取り除きます Noweb syntax references when exporting. エクスポートするときに

'`eval`'
: Expansion of Noweb syntax references in the body of the code block only before evaluating.

[5] For Noweb literate programming details, see `http://www.cs.tufts.edu/~nr/noweb/`.

In the following example,

```
#+NAME: initialization
#+BEGIN_SRC emacs-lisp
  (setq sentence "Never a foot too far, even.")
#+END_SRC

#+BEGIN_SRC emacs-lisp :noweb yes
  <<initialization>>
  (reverse sentence)
#+END_SRC
```

the second code block is expanded as

```
#+BEGIN_SRC emacs-lisp :noweb yes
  (setq sentence "Never a foot too far, even.")
  (reverse sentence)
#+END_SRC
```

Noweb insertions honor prefix characters that appear before the Noweb syntax reference. This behavior is illustrated in the following example. Because the '`<<example>>`' Noweb reference appears behind the SQL comment syntax, each line of the expanded Noweb reference is commented. With:

```
#+NAME: example
#+BEGIN_SRC text
  this is the
  multi-line body of example
#+END_SRC
```

this code block:

```
#+BEGIN_SRC sql :noweb yes
 ---<<example>>
#+END_SRC
```

expands to:

```
#+BEGIN_SRC sql :noweb yes
 ---this is the
 ---multi-line body of example
#+END_SRC
```

Since this change does not affect Noweb replacement text without newlines in them, inline Noweb references are acceptable.

This feature can also be used for management of indentation in exported code snippets. With:

```
#+NAME: if-true
#+BEGIN_SRC python :exports none
  print('do things when true')
#+end_src

#+name: if-false
#+begin_src python :exports none
```

```
  print('do things when false')
#+end_src
```

this code block:

```
#+begin_src python :noweb yes :results output
  if true:
      <<if-true>>
  else:
      <<if-false>>
#+end_src
```

expands to:

```
if true:
    print('do things when true')
else:
    print('do things when false')
```

When expanding Noweb style references, Org concatenates code blocks by matching the reference name to either the code block name or, if none is found, to the '`noweb-ref`' header argument.

For simple concatenation, set this '`noweb-ref`' header argument at the sub-tree or file level. In the example Org file shown next, the body of the source code in each block is extracted for concatenation to a pure code file when tangled.

```
#+BEGIN_SRC sh :tangle yes :noweb yes :shebang #!/bin/sh
  <<fullest-disk>>
#+END_SRC
* the mount point of the fullest disk
  :PROPERTIES:
  :header-args: :noweb-ref fullest-disk
  :END:

** query all mounted disks
#+BEGIN_SRC sh
  df \
#+END_SRC

** strip the header row
#+BEGIN_SRC sh
  |sed '1d' \
#+END_SRC

** output mount point of fullest disk
#+BEGIN_SRC sh
  |awk '{if (u < +$5) {u = +$5; m = $6}} END {print m}'
#+END_SRC
```

By default デフォルトで a newline separates 分けますう each noweb reference concatenation. To change this newline separator, edit the '`noweb-sep`' header argument.

Eventually, Org can include the results of a code block rather than its body. To that effect, その効果のためには append parentheses, p ossibly including 含む arguments, to the code block name, as shown below.

```
<<code-block-name(optional arguments)>>
```

Note 注意してください that when using 使うときには the above approach 上のアプローチを to a code block's results, the code block name set by '`NAME`' keyword is required; the reference set by '`noweb-ref`' does not work 動きません in that case. その場合には

Here これは is an example 例です that demonstrates how the exported content changes when Noweb style references are used with parentheses versus without. With:

```
#+NAME: some-code
#+BEGIN_SRC python :var num=0 :results output :exports none
  print(num*10)
#+END_SRC
```

this code block:

```
#+BEGIN_SRC text :noweb yes
  <<some-code>>
#+END_SRC
```

expands to:

```
print(num*10)
```

Below, a similar Noweb style reference is used, but with parentheses, while setting a variable '`num`' to 10:

```
#+BEGIN_SRC text :noweb yes
  <<some-code(num=10)>>
#+END_SRC
```

Note 注意してください that now the expansion contains 含んでいます the results of the code block '`some-code`', not the code block itself:

```
100
```

14.11 Library of Babel

The "Library of Babel" is a collection of code blocks. Like a function library, these code blocks can be called from other Org files. A collection of useful code blocks is available on Worg (`https://orgmode.org/worg/library-of-babel.html`). For remote code block evaluation syntax, see 節 14.4 [Evaluating Code Blocks], ページ 221.

For any user to add code to the library, first save the code in regular code blocks of an Org file, and then load the Org file with `org-babel-lob-ingest`, which これは is bound to *C-c C-v i* にバインドされていまますう

14.12 Key bindings and Useful Functions

Many common Org-mode key sequences are re-bound depending on the context. コンテキストに依存して

Active key bindings in code blocks:

Key binding	Function
C-c C-c	org-babel-execute-src-block
C-c C-o	org-babel-open-src-block-result
M-UP	org-babel-load-in-session
M-DOWN	org-babel-pop-to-session

Active key bindings in Org-mode buffer:

Key binding	Function
C-c C-v p or *C-c C-v C-p*	org-babel-previous-src-block
C-c C-v n or *C-c C-v C-n*	org-babel-next-src-block
C-c C-v e or *C-c C-v C-e*	org-babel-execute-maybe
C-c C-v o or *C-c C-v C-o*	org-babel-open-src-block-result
C-c C-v v or *C-c C-v C-v*	org-babel-expand-src-block
C-c C-v u or *C-c C-v C-u*	org-babel-goto-src-block-head
C-c C-v g or *C-c C-v C-g*	org-babel-goto-named-src-block
C-c C-v r or *C-c C-v C-r*	org-babel-goto-named-result
C-c C-v b or *C-c C-v C-b*	org-babel-execute-buffer
C-c C-v s or *C-c C-v C-s*	org-babel-execute-subtree
C-c C-v d or *C-c C-v C-d*	org-babel-demarcate-block
C-c C-v t or *C-c C-v C-t*	org-babel-tangle
C-c C-v f or *C-c C-v C-f*	org-babel-tangle-file
C-c C-v c or *C-c C-v C-c*	org-babel-check-src-block
C-c C-v j or *C-c C-v C-j*	org-babel-insert-header-arg
C-c C-v l or *C-c C-v C-l*	org-babel-load-in-session
C-c C-v i or *C-c C-v C-i*	org-babel-lob-ingest
C-c C-v I or *C-c C-v C-I*	org-babel-view-src-block-info
C-c C-v z or *C-c C-v C-z*	org-babel-switch-to-session-with-code
C-c C-v a or *C-c C-v C-a*	org-babel-sha1-hash
C-c C-v h or *C-c C-v C-h*	org-babel-describe-bindings
C-c C-v x or *C-c C-v C-x*	org-babel-do-key-sequence-in-edit-buffer

14.13 Batch Execution

Org-mode features, including 含む working with source code facilities can be invoked 実行することができます from the command line. This enables building shell scripts for batch processing, running automated system tasks, and expanding Org-mode's usefulness. Org-mode の有用さを

The sample script shows 示しています batch processing of multiple files using `org-babel-tangle`.

```
#!/bin/sh
# Tangle files with Org-mode
#
emacs -Q --batch --eval "
    (progn
      (require 'ob-tangle)
```

```
    (dolist (file command-line-args-left)
      (with-current-buffer (find-file-noselect file)
        (org-babel-tangle))))
" "$@"
```

15 Miscellaneous

15.1 Completion

Org has in-buffer completions . Unlike minibuffer completions, which これは are useful 役に立ちます for quick command interactions, Org's in-buffer completions are more suitable for content creation in Org documents. Type one or more letters and invoke the hot key to complete the text in-place. Depending on the context and the keys, コンテキストとキーに依存して、Org Org-mode は offers 提供します different types of completions. 異なる種類の補完を No minibuffer is involved. Such mode-specific hot keys have become an integral part of Emacs and Org provides 提供しています several shortcuts. 複数のショートカットを

M-TAB

Complete word at point.

- At the beginning of a headline, complete TODO keywords.
- After '\', complete TEX symbols supported by the exporter.
- After '*' の後で、complete headlines 見出しを補完します in the current buffer so that they can be used 使うことができます in search links like: このような

```
[[*find this headline]]
```

- After ':' in a headline, complete tags. Org deduces the list of tags from the 'TAGS' in-buffer option（節 6.2 [Setting Tags], ページ 62, を見てください）、the variable `org-tag-alist`, or from all tags used in the current buffer.
- After ':' and not in a headline, complete property keys. The list of keys is constructed dynamically from all keys used in the current buffer.
- After '[', complete link abbreviations（節 4.7 [Link Abbreviations], ページ 46, を見てください）。
- After '#+', complete the special keywords like 'TYP_TODO' or file-specific 'OPTIONS'. After option keyword is complete, pressing *M-TAB* again inserts 挿入します example settings 例となる設定を for this keyword.
- After 'STARTUP' keyword, complete startup items.
- When point ポイントが is anywhere else, complete 補完します dictionary words using Ispell を使って辞書の単語を

15.2 Structure Templates

With just a few keystrokes, it is possible to insert empty structural blocks, such as '#+BEGIN_SRC' . . . '#+END_SRC' といった or to wrap existing text in such a block.

C-c C-, (`org-insert-structure-template`)

Prompt プロンプトを出します for a type of block structure, and insert 挿入します the block at point. ポイントの場所に If the region is active, it それが is wrapped in the block. First prompts プロンプトを出します the user ユーザーに for keys, キーを求めて which are used to look up a structure type from the

variable below. If the key is TAB かRET かSPC であれば the user is ユーザーには prompted プロンプトを出します to enter a block type.

Available structure types are defined in `org-structure-template-alist`, see the docstring for adding or changing values.

Org Tempo expands snippets to structures defined in `org-structure-template-alist` and `org-tempo-keywords-alist`. For example, 例えば`< s TAB` は creates 作成します a code block. Enable it by customizing `org-modules` or add '`(require 'org-tempo)`' to your Emacs init file[1].

a	'`#+BEGIN_EXPORT ascii`' ... '`#+END_EXPORT`'
c	'`#+BEGIN_CENTER`' ... '`#+END_CENTER`'
C	'`#+BEGIN_COMMENT`' ... '`#+END_COMMENT`'
e	'`#+BEGIN_EXAMPLE`' ... '`#+END_EXAMPLE`'
E	'`#+BEGIN_EXPORT`' ... '`#+END_EXPORT`'
h	'`#+BEGIN_EXPORT html`' ... '`#+END_EXPORT`'
l	'`#+BEGIN_EXPORT latex`' ... '`#+END_EXPORT`'
q	'`#+BEGIN_QUOTE`' ... '`#+END_QUOTE`'
s	'`#+BEGIN_SRC`' ... '`#+END_SRC`'
v	'`#+BEGIN_VERSE`' ... '`#+END_VERSE`'

15.3 Escape Character

You may sometimes ときどき want to write 書きたいことがあります text テキストを that looks like Org syntax, but should really 本当は read as plain text. Org may use 使うことができます a specific escape character in some situations, e.g., 例えば a backslash in macros（節 12.5 [Macro Replacement], ページ 151, を見てください）。In the general case, 一般的な場合には however, しかし we suggest to use the zero width space. You can get it with one of the following:

```
C-x 8 <RET> zero width space <RET>
C-x 8 <RET> 200B <RET>
```

For example, 例えば in order to write '`[[1,2]]`' as-is in your document, you can write this, where '`X`' denotes the zero width space character:

```
[[X1,2]]
```

15.4 Speed Keys

Single keystrokes can execute custom commands in an Org file when point is on a headline. Without the extra burden of a meta or modifier key, Speed Keys can speed navigation or execute custom commands. Besides faster navigation, Speed Keys may come in handy 便利なことがあります on small mobile devices that do not have full keyboards. Speed Keys may also work on TTY devices known for their problems when entering Emacs key chords.

By default, デフォルトで Org has Speed Keys disabled. 無効にしています To activate 有効にするには Speed Keys を set the variable `org-use-speed-commands` to a non-`nil` value. `nil` でない値に To trigger トリガーをかけるには a Speed Key, point must be at the beginning of an Org headline, before any of the stars.

[1] For more information, please どうか refer 参照してください to the commentary section in '`org-tempo.el`'.

Org comes with a pre-defined list of Speed Keys. To add 追加するか or modify 変更するには Speed Keys を customize カスタマイズしてください the variable, `org-speed-commands-user`. For more details, see the variable's docstring. この変数のドキュメント文字列を見てください With Speed Keys activated, *`M-x org-speed-command-help`*, or *`?`* when point is at the beginning of an Org headline, shows 表示します currently active Speed Keys, including 含む the user-defined ones. ユーザー定義のキーを

15.5 Code Evaluation and Security Issues

Unlike plain text, running code comes with risk. Each source code block, in terms of risk, is equivalent to an executable file. Org therefore よって puts 置きます a few confirmation prompts by default. デフォルトで This is to alert the casual user from accidentally running untrusted code.

For users who do not run code blocks or write code regularly, Org's default settings should suffice. 十分なはずです However, しかし some users ユーザーもいます may want to tweak 変更したい the prompts これらのプロンプトを for fewer interruptions. To weigh the risks of automatic execution of code blocks, here are some details about code evaluation.

Org evaluates code in the following circumstances:

Source code blocks

Org evaluates source code blocks in an Org file during export. Org also evaluates a source code block with the *`C-c C-c`* key chord. Users exporting or running code blocks must load files only from trusted sources. Be wary of customizing variables that remove or alter default security measures.

`org-confirm-babel-evaluate` [ユーザオプション]

When `t` のときには、Org Org-mode は prompts プロンプトを出します the user ユーザーに for confirmation 確認を求めて before executing each code block. When `nil` のときには Org executes 実行します code blocks without prompting プロンプトを出すことなく the user ユーザーに for confirmation. When this option このオプションが is set to a custom function, Org invokes the function with these two arguments: the source code language and the body of the code block. The custom function must return either a `t` or `nil`, which これは determines 決定します if the user is prompted. プロンプトが出ます Each source code language can be handled separately through this function argument.

For example, 例えば here is how to execute ditaa code blocks without prompting: プロンプトを出すことなく

```
(defun my-org-confirm-babel-evaluate (lang body)
  (not (string= lang "ditaa")))  ;don't ask for ditaa
(setq org-confirm-babel-evaluate #'my-org-confirm-babel-evaluate)
```

Following '`shell`' and '`elisp`' links

Org Org-mode には has あります two link types that can directly 直接 evaluate 評価することができる code コードを（節 4.4 [External Links], ページ 41, を見てください）。Because such code is not visible, these links have a potential risk. Org therefore よって prompts プロンプトを出します the user ユーザーに when it encounters such links. The customization variables are:

`org-confirm-shell-link-function` [ユーザオプション]
: Function 関数です that prompts プロンプトを出します the user ユーザーに before executing 実行する前に a shell link. シェルリンクを

`org-confirm-elisp-link-function` [ユーザオプション]
: Function that prompts プロンプトを出します the user before executing an Emacs Lisp link.

Formulas in tables
: Formulas in tables（節 3.5 [The Spreadsheet], ページ 26, を見てください）are code that is evaluated either by the Calc interpreter, or by the Emacs Lisp interpreter.

15.6 Customization

Org has more than 500 variables for customization. They can be accessed through the usual *M-x org-customize* command. Or または through the Org menu: Org → Customization → Browse Org Group.

Org Org-mode には also has あります per-file settings ファイルごとの設定も for some variables（節 15.7 [In-buffer Settings], ページ 243, を見てください）。

15.7 Summary of In-Buffer Settings

In-buffer settings start with '`#+`', followed by a keyword, a colon, and then a word for each setting. Org accepts multiple settings on the same line. Org also accepts multiple lines for a keyword. This manual describes these settings throughout. A summary follows here.

C-c C-c activates any changes to the in-buffer settings. Closing and reopening the Org file in Emacs also activates the changes.

'`#+ARCHIVE: %s_done`'
: Sets the archive location of the agenda file. The corresponding variable is `org-archive-location`.

'`#+CATEGORY`'
: Sets the category of the agenda file, which これは applies 適用されます to the entire document. ドキュメント全体に対して

'`#+COLUMNS: %25ITEM ...`'
: Set the default format for columns view. This format applies when columns view is invoked in locations where no '`COLUMNS`' property applies.

'`#+CONSTANTS: name1=value1 ...`'
: Set 設定します file-local values ファイルに対してローカルな値を for constants 定数に対する that table formulas 表の数式が can use. 使うことができる This line この行は sets 設定します the local variable `org-table-formula-constants-local`. The global version of this variable この変数の is `org-table-formula-constants` でせいう。

'`#+FILETAGS: :tag1:tag2:tag3:`'
: Set tags that all entries in the file inherit from, including 含む the top-level entries.

‘`#+LINK: linkword replace`’
: Each line specifies 指定します one abbreviation for one link. Use 使ってください multiple ‘`LINK`’ keywords for more, see 節 4.7 [Link Abbreviations], ページ 46. The corresponding variable is `org-link-abbrev-alist`.

‘`#+PRIORITIES: highest lowest default`’
: This line sets the limits and the default for the priorities. All three must be either letters A--Z or numbers 0--9. The highest priority must have a lower ASCII number than the lowest priority.

‘`#+PROPERTY: Property_Name Value`’
: This line sets a default inheritance value for entries in the current buffer, most 主に useful 役に立ちます for specifying the allowed values of a property.

‘`#+SETUPFILE: file`’
: The setup file or a URL pointing to such file is for additional in-buffer settings. Org loads this file and parses it for any settings in it only when Org opens the main file. If URL is specified, the contents are downloaded and stored in a temporary file cache. `C-c C-c` on the settings line parses and loads the file, and also resets the temporary file cache. Org also parses and loads the document during normal exporting process. Org parses the contents of this document as if it was included in the buffer. It can be another Org file. To visit 訪れるには the file ---not a URL---use `C-c '` while point is on the line with the file name.

‘`#+STARTUP:`’
: Startup options Org uses when first visiting a file.

  The first set of options deals with the initial visibility of the outline tree. The corresponding variable for global default settings is `org-startup-folded` with a default value of `t`, which これは is the same as `overview` と同じです

  ‘`overview`’
  : Top-level headlines only.

  ‘`content`’ All headlines.

  ‘`showall`’ No folding on any entry.

  ‘`showeverything`’
  : Show even drawer contents.

  Dynamic virtual indentation is controlled by the variable `org-startup-indented`[2].

  ‘`indent`’ Start with `org-indent-mode` turned on.

  ‘`noindent`’
  : Start with `org-indent-mode` turned off.

  Aligns tables consistently upon visiting a file. The corresponding variable is `org-startup-align-all-tables` with `nil` as default value.

  ‘`align`’ Align all tables.

[2] Note 注意してください that `org-indent-mode` also sets the `wrap-prefix` property, such that `visual-line-mode` (or purely setting `word-wrap`) wraps long lines (including 含む headlines) correctly indented.

‘noalign’ Do not align tables on startup.

Shrink table columns with a width cookie. The corresponding variable is org-startup-shrink-all-tables with nil as default value.

When visiting a file, inline images can be automatically 自動的に displayed. 表示することができます The corresponding variable is org-startup-with-inline-images, with a default value nil to avoid delays when visiting a file.

‘inlineimages’
Show inline images.

‘noinlineimages’
Do not show inline images on startup.

Logging the closing and reopening of TODO items and clock intervals can be configured 設定することができます using these options（~org-log-done~ とorg-log-note-clock-out とorg-log-repeat を見てください）。

‘logdone’ Record a timestamp when an item is marked DONE.

‘lognotedone’
Record timestamp and a note when DONE.

‘nologdone’
Do not record when items are marked DONE.

‘logrepeat’
Record a time when reinstating a repeating item.

‘lognoterepeat’
Record a note when reinstating a repeating item.

‘nologrepeat’
Do not record when reinstating repeating item.

‘lognoteclock-out’
Record a note when clocking out.

‘nologlognoteclock-out’
Do not record a note when clocking out.

‘logreschedule’
Record a timestamp when scheduling time changes.

‘lognotereschedule’
Record a note when scheduling time changes.

‘nologreschedule’
Do not record when a scheduling date changes.

‘logredeadline’
Record a timestamp when deadline changes.

‘lognoteredeadline’
Record a note when deadline changes.

‘nologredeadline’
Do not record when a deadline date changes.

‘logrefile’
Record a timestamp when refiling.

‘lognoterefile’
Record a note when refiling.

‘nologrefile’
Do not record when refiling.

Here これらは are the options オプションです for hiding leading stars in outline headings, and for indenting outlines. The corresponding variables are `org-hide-leading-stars` and `org-odd-levels-only`, both with a default setting `nil` (meaning 意味しますう‘`showstars`’ and ‘`oddeven`’).

‘hidestars’
Make all but one of the stars starting a headline invisible.

‘showstars’
Show all stars starting a headline.

‘indent’ Virtual indentation according 従って to outline level.

‘noindent’
No virtual indentation according 従って to outline level.

‘odd’ Allow only odd outline levels (1, 3, . . .).

‘oddeven’ Allow all outline levels.

To turn on オンにするには custom format overlays over timestamps (variables `org-put-time-stamp-overlays` and `org-time-stamp-overlay-formats`), use:

‘customtime’
Overlay custom time format.

The following options influence the table spreadsheet (variable `constants-unit-system`).

‘constcgs’
‘`constants.el`’ should use 使うべきです the c-g-s 単位系を

‘constSI’ ‘`constants.el`’ は should use 使うべきです the SI 単位系を

To influence footnote settings, use the following keywords. The corresponding variables are `org-footnote-define-inline`, `org-footnote-auto-label`, and `org-footnote-auto-adjust`.

‘fninline’
Define footnotes inline.

‘fnnoinline’
Define footnotes in separate section.

‘fnlocal’ Define footnotes near first reference, but not inline.

‘`fnprompt`’
: Prompt プロンプトを出します for footnote labels.

‘`fnauto`’
: Create 作成します‘`[fn:1]`’-like labels automatically 自動的に(default).（デフォルト）。

‘`fnconfirm`’
: Offer automatic label for editing or confirmation.

‘`fnadjust`’
: Automatically 自動的に renumber and sort footnotes.

‘`nofnadjust`’
: Do not renumber and sort automatically. 自動的に

To hide 隠すには blocks on startup, use these keywords. The corresponding variable is `org-hide-block-startup`.

‘`hideblocks`’
: Hide all begin/end blocks on startup.

‘`nohideblocks`’
: Do not hide blocks on startup.

The display of entities as UTF-8 characters is governed by the variable `org-pretty-entities` and the keywords

‘`entitiespretty`’
: Show entities as UTF-8 characters where possible.

‘`entitiesplain`’
: Leave entities plain.

‘`#+TAGS: TAG1(c1) TAG2(c2)`’
: These lines (several such lines are allowed) specify the valid tags in this file, and (potentially) the corresponding *fast tag selection* keys. The corresponding variable is `org-tag-alist`.

‘`#+TODO:`’
‘`#+SEQ_TODO:`’
‘`#+TYP_TODO:`’
: These lines set the TODO keywords and their interpretation in the current file. The corresponding variable is `org-todo-keywords`.

15.8 The Very Busy *C-c C-c* Key

The *C-c C-c* key in Org serves many purposes depending on the context. コンテキストに依存して It is probably the most over-worked, multi-purpose key combination in Org. Its uses are well documented throughout this manual, but here is a consolidated list for easy reference.

- If any highlights shown in the buffer from the creation of a sparse tree, or from clock display, remove 取り除きます such highlights.
- If point ポイントが is in one of the special ‘`KEYWORD`’ lines, scan the buffer for these lines and update the information. Also また reset the Org file cache used to temporary store the contents of URLs used as values for keywords like ‘`SETUPFILE`’.

- If point ポイントが is inside a table, realign the table. The table realigns even if automatic table editor is turned off.
- If point ポイントが is on a '`TBLFM`' keyword, re-apply the formulas to the entire table.
- If the current buffer is a capture buffer, close the note and file it. With a prefix argument, 1 つの前置引数といっしょだと、also jump to the target location after saving the note.
- If point ポイントが is on a '`<<<target>>>`', update 更新します radio targets and corresponding links in this buffer.
- If point ポイントが is on a property line or at the start or end of a property drawer, offer property commands.
- If point ポイントが is at a footnote reference, go to the corresponding definition, and *vice versa*.
- If point ポイントが is on a statistics cookie, update it. それを更新します
- If point ポイントが is in a plain list item with a checkbox, toggle the status of the checkbox.
- If point ポイントが is on a numbered item in a plain list, renumber the ordered list.
- If point ポイントが is on the '`#+BEGIN`' line of a dynamic block, the block is updated.
- If point ポイントが is at a timestamp, fix the day name in the timestamp.

15.9 A Cleaner Outline View

Org's default outline with stars and no indents can become too cluttered for short documents. For *book-like* long documents, the effect is not as noticeable. 顕著ではありません Org Org-mode は provides 提供しています an alternate stars and indentation scheme, as shown on the right in the following table. It uses only one star and indents text to line with the heading:

```
* Top level headline             |    * Top level headline
** Second level                  |      * Second level
*** Third level                  |        * Third level
    some text                    |          some text
*** Third level                  |        * Third level
    more text                    |          more text
* Another top level headline     |    * Another top level headline
```

To turn this mode on, このモードをオンにするには use the minor mode, `org-indent-mode`. Text lines that are not headlines are prefixed with spaces to vertically align with the headline text[3].

To make more horizontal space, the headlines are shifted by two stars. This can be configured 設定することができます by the `org-indent-indentation-per-level` 変数によって Only one star on each headline is visible, the rest are masked with the same font color as the background.

[3] The `org-indent-mode` also sets the `wrap-prefix` correctly for indenting and wrapping long lines of headlines or text. This minor mode handles `visual-line-mode` and directly 直接 applied settings through `word-wrap`.

Note 注意してください that turning on `org-indent-mode` を sets `org-hide-leading-stars` を to `t` and `org-adapt-indentation` to `nil`.

To globally turn on グローバルにオンにするには`org-indent-mode` を for all files, customize カスタマイズしてください the variable 変数`org-startup-indented` を

To turn on オンにするには indenting for individual files, use 使ってください`STARTUP`' キーワードを as follows: 次のようにしたえ

```
#+STARTUP: indent
```

Indent on startup makes Org use hard spaces to align text with headings as shown in examples below.

Indentation of text below headlines

Indent text to align with the headline.

```
*** Third level
    more text, now indented
```

Org supports サポートしています this これを with paragraph filling, line wrapping, and structure editing, preserving or adapting the indentation as appropriate[4].

Hiding leading stars

Org can make leading stars invisible. 不可視にすることができあますう For global preference, グローバルな設定に対しては、configure 設定してください。 the 変数`org-hide-leading-stars` を For per-file preference, ファイルごとの設定に対しては use 使ってください these file '`STARTUP`' options:

```
#+STARTUP: hidestars
#+STARTUP: showstars
```

With stars hidden, the tree is shown as:

```
* Top level headline
 * Second level
  * Third level
  ...
```

Because Org makes the font color the same as the background color to hide to stars, sometimes ときどき`org-hide` face may need 必用かもしれません tweaking to get the effect right. For some black and white combinations, `grey90` on a white background might mask the stars better.

Odd levels

Using stars for only odd levels, 1, 3, 5, ..., can also clean up the clutter. This removes 取り除きます two stars from each level[5]. For Org to properly handle this cleaner structure during edits and exports, configure 設定してください。 the variable `org-odd-levels-only`. To set 設定するには this これを per-file, ファイルごとに use 使ってください either one of the following lines:

```
#+STARTUP: odd
#+STARTUP: oddeven
```

[4] Also see 見てください。the variable 変数`org-adapt-indentation` も

[5] with the command *`M-x orgtbl-insert-radio-table`*, which これは prompts プロンプトを出します for a table name. For example, 例えば if '`salesfigures`' が is the name, その名前であれば the template このテンプレートは inserts: 挿入します

To switch between single and double stars layouts, use *M-x org-convert-to-odd-levels* と*M-x org-convert-to-oddeven-levels*.

15.10 Dynamic Headline Numbering

The Org Num minor mode, toggled トグルされる with *M-x org-num-mode*, displays 表示します on top of headlines. It also updates numbering automatically 自動的に upon changes to the structure of the document.

By default, デフォルトで all headlines 全ての見出しが are numbered. You can limit numbering to specific headlines according 従って to their level, tags, 'COMMENT' keyword, or 'UNNUMBERED' property. Set `org-num-max-level`, `org-num-skip-tags`, `org-num-skip-commented`, `org-num-skip-unnumbered`, or `org-num-skip-footnotes` accordingly.

If `org-num-skip-footnotes` が`nil` でない値の場合には footnotes sections（節 2.10 [Creating Footnotes], ページ 17, を見てください）are not numbered either.

You can control how the numbering is displayed by setting `org-num-face` と`org-num-format-function` を設定することによって

15.11 Using Org on a TTY

Org provides 提供しています alternative key bindings for TTY and modern mobile devices that cannot perform movement commands on point and key bindings with modifier keys. Some of these workarounds may be more cumbersome than necessary. 必用委譲にやっかいかもしれません Users should look into customizing these further based on their usage needs. For example, 例えば the normal 通常の*S-<cursor>* は for editing timestamp might be better 良いかもしれませわい with *C-c .* chord.

デフォルト	**Alternative 1**	**Speed key**	**Alternative 2**
S-TAB	*C-u TAB*	*C*	
M-LEFT	*C-c C-x l*	*l*	*Esc LEFT*
M-S-LEFT	*C-c C-x L*	*L*	
M-RIGHT	*C-c C-x r*	*r*	*Esc RIGHT*
M-S-RIGHT	*C-c C-x R*	*R*	
M-UP	*C-c C-x u*		*Esc UP*
M-S-UP	*C-c C-x U*	*U*	
M-DOWN	*C-c C-x d*		*Esc DOWN*
M-S-DOWN	*C-c C-x D*	*D*	
S-RET	*C-c C-x c*		
M-RET	*C-c C-x m*		*Esc RET*
M-S-RET	*C-c C-x M*		
S-LEFT	*C-c LEFT*		
S-RIGHT	*C-c RIGHT*		
S-UP	*C-c UP*		
S-DOWN	*C-c DOWN*		
C-S-LEFT	*C-c C-x LEFT*		
C-S-RIGHT	*C-c C-x RIGHT*		

15.12 Context Dependent Documentation

`C-c C-x C-i` in an Org file tries to open a suitable section of the Org manual Org-mode のマニュアルの depending on 依存して the syntax 文法に at point. ポイントにある For example, 例えば using it それを使うと、on a headline 見出しの上で displays 表示します。"Document Structure" セクションを

`q` closes the Info window.

15.13 Interaction with Other Packages

Org's compatibility and the level of interaction with other Emacs packages are documented here.

15.13.1 Packages that Org cooperates with

'`calc.el`' by Dave Gillespie
: Org uses 使います the Calc package for implementing spreadsheet functionality in its tables（節 3.5 [The Spreadsheet], ページ 26, を見てください）。Org also uses Calc for embedded calculations. See 節 "Embedded Mode" in `calc`.

'`constants.el`' by Carsten Dominik
: Org Org-mode は can use 使うことができます names 名前を for constants 定数に対して in formulas in tables. 表の中の数式の中で Org can also use 使うこともできます calculation suffixes for units, such as '`M`' といった for '`Mega`' を意味する For a standard collection of such constants, install the '`constants`' package. Install version 2.0 of this package, available at `http://www.astro.uva.nl/~dominik/Tools`. Org checks if the function `constants-get` has been autoloaded. Installation instructions are in the file '`constants.el`'.

'`cdlatex.el`' by Carsten Dominik
: Org-mode can make use of the CDLaTeX package to efficiently enter LaTeX fragments into Org files. See 節 11.5.3 [CDLaTeX mode], ページ 140.

'`imenu.el`' by Ake Stenhoff and Lars Lindberg
: Imenu は creates 作成します dynamic menus based on an index of items in a file. Org-mode supports サポートしています Imenu menus. Enable it with a mode hook as follows:

```
(add-hook 'org-mode-hook
          (lambda () (imenu-add-to-menubar "Imenu")))
```

: By default デフォルトで the index is two levels deep---you can modify the depth using 使って the option `org-imenu-depth`.

'`speedbar.el`' by Eric M. Ludlam
: Speedbar パッケージは creates 作成します a special Emacs frame for displaying files and index items in files. Org-mode は supports サポートしています Speedbar; users can drill into Org files directly 直接 from the Speedbar から The `<` in the Speedbar frame tweaks the agenda commands to that file or to a subtree.

‘`table.el`’ by Takaaki Ota

Complex ASCII tables with automatic line wrapping, column- and row-spanning, and alignment can be created using the Emacs table package by Takaaki Ota. Org-mode recognizes such tables and exports them properly. *C-c '* to edit these tables in a special buffer, much like Org’s code blocks. Because of interference 干渉のために with other Org-mode functionality, Takaaki Ota tables cannot be edited 編集することはできません directly 直接 in the Org buffer.

C-c ' (`org-edit-special`)

Edit a ‘`table.el`’ table. Works when point is in a ‘`table.el`’ table.

C-c ~ (`org-table-create-with-table.el`)

Insert a ‘`table.el`’ table. If there is already a table at point, this command converts 変換します it それを between the ‘`table.el`’ format and the Org-mode format. See the documentation string of the command `org-convert-table` for the restrictions under which this is possible.

15.13.2 Packages that conflict with Org-mode

In Emacs, `shift-selection-mode` combines motions of point with shift key to enlarge regions. Emacs sets this mode by default. デフォルトで This これは conflicts 衝突します with Org’s use of *S-<cursor>* commands to change timestamps, TODO keywords, priorities, and item bullet types, etc. Since *S-<cursor>* commands outside of specific contexts do not do anything, 何もしないので Org Org-mode は offers 提供します the variable `org-support-shift-select` for customization. Org-mode accommodates shift selection by (i) making it available outside of the special contexts where special commands apply, and (ii) extending an existing active region even if point moves across a special context.

‘`cua.el`’ by Kim F. Storm

Org key bindings conflict with *S-<cursor>* keys used by CUA mode. For Org to relinquish these bindings to CUA mode, configure 設定してください。the variable `org-replace-disputed-keys`. When set, Org moves the following key bindings in Org files, and in the agenda buffer---but not during date selection.

S-UP ⇒ *M-p*	*S-DOWN* ⇒ *M-n*
S-LEFT ⇒ *M--*	*S-RIGHT* ⇒ *M-+*
C-S-LEFT ⇒ *M-S--*	*C-S-RIGHT* ⇒ *M-S-+*

Yes, these are unfortunately more difficult to remember. If you want to have other replacement keys, look at the variable `org-disputed-keys`.

‘`ecomplete.el`’ by Lars Magne Ingebrigtsen

Ecomplete は provides 提供しています"electric" address completion in address header lines in message buffers. Sadly Orgtbl mode cuts Ecomplete’s power supply: no completion happens when Orgtbl mode is enabled in message buffers while entering text in address header lines. If one wants to use ecomplete one should *not* follow 従うべきではありません the advice to automagically turn on Orgtbl mode in message buffers（節 3.4 [Orgtbl Mode], ページ 25, を見てください）、but instead かわりに---after filling in the message headers ---turn on Orgtbl mode manually when needed in the messages body.

‘filladapt.el’ by Kyle Jones
: Org-mode tries to do the right thing when filling paragraphs, list items and other elements. Many users reported problems using both ‘filladapt.el’ and Org-mode, so a safe thing to do is to disable filladapt like this: このようにして

```
(add-hook 'org-mode-hook 'turn-off-filladapt-mode)
```

‘viper.el’ by Michael Kifer
: Viper uses *C-c /* and therefore makes this key not access the corresponding Org-mode command org-sparse-tree. You need to find 見つける必用があります another key for this command, or override the key in viper-vi-global-user-map with

```
(define-key viper-vi-global-user-map "C-c /" 'org-sparse-tree)
```

‘windmove.el’ by Hovav Shacham
: This package also uses the *S-<cursor>* keys, so everything written in the paragraph above about CUA mode also applies here. If you want to make the windmove function active in locations where Org-mode does not have special functionality on *S-<cursor>*, add this to your configuration:

```
;; Make windmove work in Org-mode:
(add-hook 'org-shiftup-final-hook 'windmove-up)
(add-hook 'org-shiftleft-final-hook 'windmove-left)
(add-hook 'org-shiftdown-final-hook 'windmove-down)
(add-hook 'org-shiftright-final-hook 'windmove-right)
```

‘yasnippet.el’
: The way Org-mode binds the TAB key (binding to [tab] instead かわりに of "\t") overrules YASnippet’s access to this key. The following code fixed this problem:

```
(add-hook 'org-mode-hook
          (lambda ()
            (setq-local yas/trigger-key [tab])
            (define-key yas/keymap [tab] 'yas/next-field-or-maybe-expand)))
```

The latest version of YASnippet does not play well with Org-mode. If the above code does not fix the conflict, start by defining 定義することによって the following function:

```
(defun yas/org-very-safe-expand ()
  (let ((yas/fallback-behavior 'return-nil)) (yas/expand)))
```

Then, tell Org-mode to use that function:

```
(add-hook 'org-mode-hook
          (lambda ()
            (make-variable-buffer-local 'yas/trigger-key)
            (setq yas/trigger-key [tab])
            (add-to-list 'org-tab-first-hook 'yas/org-very-safe-expand)
            (define-key yas/keymap [tab] 'yas/next-field)))
```

15.14 Org Crypt

Org Crypt encrypts the text of an entry, but not the headline, or properties. Behind the scene, it uses the Emacs EasyPG library to encrypt and decrypt files.

Any text below a headline that has a ‘`crypt`’ tag is automatically 自動的に encrypted when the file is saved. To use a different tag, 異なるタグを使うには、customize カスタマイズしてください the `org-crypt-tag-matcher` 設定を

Here これは is a suggestion 提案です for Org Crypt settings in Emacs init file:

```
(require 'org-crypt)
(org-crypt-use-before-save-magic)
(setq org-tags-exclude-from-inheritance '("crypt"))

(setq org-crypt-key nil)
;; GPG key to use for encryption
;; Either the Key ID or set to nil to use symmetric encryption.

(setq auto-save-default nil)
;; Auto-saving does not cooperate with org-crypt.el: so you need to
;; turn it off if you plan to use org-crypt.el quite often.  Otherwise,
;; you'll get an (annoying) message each time you start Org.

;; To turn it off only locally, you can insert this:
;;
;; # -*- buffer-auto-save-file-name: nil; -*-
```

It’s possible to use different keys for different headings by specifying 指定することによって the respective key as property ‘`CRYPTKEY`’, e.g.: 例えば

```
* Totally secret :crypt:
  :PROPERTIES:
  :CRYPTKEY: 0x0123456789012345678901234567890123456789
  :END:
```

Excluding the ‘`crypt`’ tag from inheritance prevents already encrypted text from being encrypted again.

15.15 Org Mobile

Org Mobile is a protocol for synchronizing Org files between Emacs and other applications, e.g., 例えば on mobile devices. It enables offline-views and capture support for an Org-mode system that is rooted on a "real" computer. The external application can also record changes to existing entries.

This appendix describes Org’s support for agenda view formats compatible with Org Mobile. It also describes synchronizing changes, such as to notes, between the mobile application and the computer.

To change 変更するには tags and TODO states in the mobile application, モバイルアプリケーションの名あで first customize カスタマイズしてください the variables 変数`org-todo-keywords` と`org-tag-alist` を These should cover カバーするべきです all the important tags and TODO keywords, even if Org files use 使うとしても only some of them.

Though the mobile application is expected to support in-buffer settings, it is required to understand TODO states *sets*（節 5.2.5 [Per-file keywords], ページ 53, を見てください）and *mutually exclusive* tags（節 6.2 [Setting Tags], ページ 62, を見てください）only for those set in these variables.

15.15.1 Setting up the staging area

The mobile application needs 必用があります access アクセスする to a file directory on a server[6] to interact with Emacs. Pass its location through the `org-mobile-directory` variable. If you can mount that directory locally just set the variable to point to that directory:

```
(setq org-mobile-directory "~/orgmobile/")
```

Aternatively, by using 使うことによって TRAMP（`tramp` を見てください）、`org-mobile-directory` may point 指すことができあす to a remote directory accessible through, for example, 例えば SSH, SCP, or DAVS:

```
(setq org-mobile-directory "/davs:user@remote.host:/org/webdav/")
```

With a public server, consider encrypting the files. Org also requires OpenSSL installed on the local computer. To turn on encryption, set the same password in the mobile application and in Emacs. Set the password in the variable `org-mobile-use-encryption`[7]. Note 注意してください that even after the mobile application encrypts the file contents, the file name remains visible on the file systems of the local computer, the server, and the mobile device.

15.15.2 Pushing to the mobile application

The command `org-mobile-push` copies files listed in `org-mobile-files` into the staging area. Files include agenda files (as listed in `org-agenda-files`). Customize カスタマイズしてください`org-mobile-files` を to add other files. File names are staged with paths relative to `org-directory`, so all files should be inside this directory[fn:152].

Push プッシュは creates 作成します a special Org file '`agendas.org`' with custom agenda views defined by the user[8].

Finally, 最後に Org writes the file '`index.org`', containing 含んでいる links to other files. The mobile application reads this file first from the server to determine what other files to download for agendas. For faster downloads, it is expected to only read files whose checksums[9] have changed.

---

[6] For a server to host files, consider using a WebDAV server, such as Nextcloud (`https://nextcloud.com`) といった Additional help is at this FAQ entry (`https://orgmode.org/worg/org-faq.html#mobileorg_webdav`).

[7] If Emacs is configured for safe storing of passwords, then configure 設定してください。the variable 変数~org-mobile-encryption-password~を; please どうか read 読んでください the docstring of that variable. その変数のドキュメント文字列を

[8] While creating 作成する合いですあに the agendas, Org-mode は forces 強制します'`ID`' プロパティを on all referenced entries, so that these entries can be uniquely identified if Org Mobile flags them for further action. To avoid 避けるには setting properties configure 設定してください。the variable 変数`org-mobile-force-id-on-agenda-items` を`nil` い Org-mode は then そうすると relies on outline paths, assuming they are unique.

[9] Checksums チェックサムは are stored automatically 自動的に保存されます in the file '`checksums.dat`'.

15.15.3 Pulling from the mobile application

The command `org-mobile-pull` synchronizes changes with the server. More specifically, it first pulls the Org files for viewing. It then appends captured entries and pointers to flagged or changed entries to the file '`mobileorg.org`' on the server. Org ultimately integrates its data in an inbox file format, through the following steps:

1. Org moves all entries found in '`mobileorg.org`'[10] and appends them to the file pointed to by the variable 変数`org-mobile-inbox-for-pull`. It should reside あるべきではありません neither in the staging area nor on the server. Each captured entry and each editing event is a top-level entry in the inbox file.
2. After moving the entries, Org processes changes to the shared files. Some of them are applied directly 直接 and without user interaction. Examples include changes to tags, TODO state, headline and body text. Entries requiring further action are tagged as '`FLAGGED`'. Org marks entries with problems with an error message in the inbox. They have to be resolved manually.
3. Org generates 生成します an agenda view for flagged entries for user intervention to clean up. For notes stored in flagged entries, Org displays 表示します them in the echo area when point is on the corresponding agenda item.

   `?` Pressing 押すと`?` を displays 表示します the entire flagged note in another window. Org also pushes it to the kill ring. To store flagged note as a normal note, use *`? z C-y C-c C-c`*. Pressing `?` twice does these things: first it removes 取り除きます the '`FLAGGED`' tag; second, it removes 取り除きます the flagged note from the property drawer; third, it signals that manual editing of the flagged entry is now finished.

From the agenda dispatcher, `?` returns to the view to finish processing flagged entries. Note 注意してください that these entries may not be the most recent since the mobile application searches files that were last pulled. To get an updated agenda view with changes since the last pull, pull again.

15.16 Org Syntax

A reference document providing a formal description of Org's syntax is available as a draft on Worg (`https://orgmode.org/worg/dev/org-syntax.html`), written and maintained by Nicolas Goaziou. It defines 定義しています Org's core internal concepts such as "headlines", "sections", "affiliated keywords", "(greater) elements" and "objects". Each part of an Org document belongs to one of the previous categories.

To explore 探求するには the abstract structure of an Org buffer, run this in a buffer:

```
M-: (org-element-parse-buffer) <RET>
```

It outputs a list containing 含んでいる the buffer's content represented as an abstract structure. The export engine relies on the information stored in this list. Most interactive commands ---e.g., 例えば for structure editing---also rely on the syntactic meaning of the surrounding context.

You can probe the syntax of your documents with the command

```
M-x org-lint <RET>
```

[10] The file will be empty after this operation.

It runs a number of checks to find common mistakes. It それは then displays 表示します their location in a dedicated buffer, along with a description 説明とともに and a "trust level", since false-positive are possible. From there, you can operate on the reports with the following keys:

C-j, TAB	Display the offending line
RET	Move point to the offending line
g	Check the document again
h	Hide all reports from the same checker
i	Also remove them from all subsequent checks
S	Sort reports by the column at point

付録 A Hacking

This appendix describes some ways a user can extend the functionality of Org.

A.1 Hooks

Org has a large number of hook variables for adding functionality. This appendix illustrates using a few. A complete list of hooks with documentation is maintained by the Worg project at `https://orgmode.org/worg/doc.html#hooks`.

A.2 Add-on Packages

Various authors wrote a large number of add-on packages for Org.

These packages are not part of Emacs, but they are distributed as contributed packages with the separate release available at `https://orgmode.org`. See the '`contrib/README`' file in the source code directory for a list of contributed files. Worg page with more information is at: `https://orgmode.org/worg/org-contrib/`.

A.3 Adding Hyperlink Types

Org has many built-in hyperlink types（章 4 [Hyperlinks], ページ 40, を見てください）、and an interface for adding new link types. The following example shows 示しています the process プロセスを of adding 追加する Org links Org-mode のリンクを to Unix man pages, which これは look like this このように見えあす:

```
[[man:printf][The printf manual]]
```

The following '`org-man.el`' file implements it

```
;;; org-man.el - Support for links to man pages in Org-mode
(require 'org)

(org-link-set-parameters "man"
                         :follow org-man-command
                         :export #'org-man-export
                         :store #'org-man-store-link)

(defcustom org-man-command 'man
  "The Emacs command to be used to display a man page."
  :group 'org-link
  :type '(choice (const man) (const woman)))

(defun org-man-store-link ()
  "Store a link to a man page."
  (when (memq major-mode '(Man-mode woman-mode))
    ;; This is a man page, we do make this link.
    (let* ((page (org-man-get-page-name))
           (link (concat "man:" page))
           (description (format "Man page for %s" page)))
      (org-store-link-props
```

```
        :type "man"
        :link link
        :description description))))

(defun org-man-get-page-name ()
  "Extract the page name from the buffer name."
  ;; This works for both `Man-mode' and `woman-mode'.
  (if (string-match " \\(\\S-+\\)\\*" (buffer-name))
      (match-string 1 (buffer-name))
    (error "Cannot create link to this man page")))

(defun org-man-export (link description format)
  "Export a man page link from Org files."
  (let ((path (format "http://man.he.net/?topic=%s&section=all" link))
        (desc (or description link)))
    (pcase format
      (`html (format "<a target=\"_blank\" href=\"%s\">%s</a>" path desc))
      (`latex (format "\\href{%s}{%s}" path desc))
      (`texinfo (format "@uref{%s,%s}" path desc))
      (`ascii (format "%s (%s)" desc path))
      (t path))))

(provide 'org-man)
;;; org-man.el ends here
```

To activate 有効にするには links リンクを to man pages in Org, enter this in the Emacs init file:

```
(require 'org-man)
```

A review of ‘`org-man.el`’:

1. First, ‘`(require 'org)`’ ensures ‘`org.el`’ is loaded.
2. Then `org-link-set-parameters` defines 定義します a new link type with ‘`man`’ prefix and associates functions for following, exporting and storing such links. See the variable 変数`org-link-parameters` を for a complete list of possible associations.
3. The rest of the file implements necessary variables and functions.

   For example, 例えば`org-man-store-link` には is responsible for storing a link when `org-store-link`（節 4.5 [Handling Links], ページ 43, を見てください）is called from a buffer displaying a man page. It first checks if the `major-mode` is appropriate. If check fails, the function returns `nil`, which これは means 意味します it isn't responsible for creating a link to the current buffer. Otherwise そうでなければ the function この関数は makes a link string by combining the ‘`man:`’ prefix with the man topic. It also provides 提供しています a default description. The function `org-insert-link` can insert 挿入することができます it back それを into an Org buffer later on. 青で<

A.4 Adding Export Back-ends

Org's export engine makes it easy for writing new back-ends. The framework on which the engine was built makes it easy to derive new back-ends from existing ones.

The two main entry points to the export engine are: `org-export-define-backend` and `org-export-define-derived-backend`. To grok these functions, see '`ox-latex.el`' for an example of defining a new back-end from scratch, and '`ox-beamer.el`' for an example of deriving from an existing engine.

For creating a new back-end from scratch, first set its name as a symbol in an alist consisting of elements and export functions. To make the back-end visible to the export dispatcher, set `:menu-entry` keyword. For export options specific to this back-end, set the `:options-alist`.

For creating a new back-end from an existing one, set `:translate-alist` to an alist of export functions. This alist replaces the parent back-end functions.

For complete documentation, see the Org Export Reference on Worg (`https://orgmode.org/worg/dev/org-export-reference.html`) を見てください。

A.5 Tables in Arbitrary Syntax

Due to Org's success in handling tables with Orgtbl, a frequently requested feature is the use of Org's table functions in other modes, e.g., 例えばLaTeX. This would be hard to do in a general way without complicated customization nightmares. Moreover, that would take Org away from its simplicity roots that Orgtbl has proven. There is, however, しかし an alternate approach to accomplishing the same.

This approach involves implementing a custom *translate* function that operates on a native Org *source table* to produce a table in another format. This strategy would keep the excellently working Orgtbl simple and isolate complications, if any, confined to the translate function. To add 追加するには more alien table formats, we just add more translate functions. Also また the burden of developing custom translate functions for new table formats is in the hands of those who know those formats best.

A.5.1 Radio tables

Radio tables are target locations for translated tables that are not near their source. Org finds the target location and inserts 挿入します the translated table.

The key to finding the target location is the magic words '`BEGIN/END RECEIVE ORGTBL`'. They have to appear as comments in the current mode. If the mode is C, then:

```
/* BEGIN RECEIVE ORGTBL table_name */
/* END RECEIVE ORGTBL table_name */
```

At the location of source, Org Org-mode は needs 必用です a special line to direct Orgtbl to translate and to find the target for inserting the translated table. 例です:

```
#+ORGTBL: SEND table_name translation_function arguments ...
```

'`table_name`' is the table's reference name, which これは is also used 使われます in the receiver lines, and the '`translation_function`' is the Lisp function that translates. This line, in addition, 加えて may also contain 含むこともできます alternating key and value arguments at the end. The translation function gets these values as a property list. A few standard parameters are already recognized and acted upon before the translation function is called:

'`:skip N`' Skip スキップします the first N lines of the table. Hlines do count; include them if they are to be skipped.

‘:skipcols (n1 n2 ...)’
: List of columns to be skipped. First Org automatically 自動的に discards 捨てます columns with calculation marks and then sends the table to the translator function, which これは then skips スキップします columns as specified in ‘skipcols’.

To keep the source table intact そのままで in the buffer without being disturbed when the source file is compiled or otherwise そうでなければ being worked on, use 使ってください one of these strategies:

- Place the table in a block comment. For example, 例えば in C mode you could wrap the table between ‘/*’ and ‘*/’ lines.
- Put the table after an "end" statement. For example 例えば\bye in TEX and \end{document} in LATEX.
- Comment and un-comment each line of the table during edits. The *M-x orgtbl-toggle-comment* command makes toggling easy.

A.5.2 A LATEX example of radio tables

To wrap ラップするには a source table in LATEX, use the ‘comment’ environment provided by ‘comment.sty’[1]. To activate it, それを有効にするには put \usepackage{comment} in the document header. Orgtbl mode inserts 挿入します a radio table skeleton

```
% BEGIN RECEIVE ORGTBL salesfigures
% END RECEIVE ORGTBL salesfigures
\begin{comment}
#+ORGTBL: SEND salesfigures orgtbl-to-latex
| | |
\end{comment}
```

The line ‘#+ORGTBL: SEND’ tells Orgtbl mode to use the function orgtbl-to-latex to convert the table to LATEX format, then insert the table at the target (receive) location named ‘salesfigures’. Now the table is ready for data entry. It can even use spreadsheet features[2]:

```
% BEGIN RECEIVE ORGTBL salesfigures
% END RECEIVE ORGTBL salesfigures
\begin{comment}
#+ORGTBL: SEND salesfigures orgtbl-to-latex
| Month | Days | Nr sold | per day |
|-------+------+---------+---------|
| Jan   |   23 |      55 |     2.4 |
| Feb   |   21 |      16 |     0.8 |
| March |   22 |     278 |    12.6 |
#+TBLFM: $4=$3/$2;%.1f
```

[1] https://www.ctan.org/pkg/comment

[2] If the ‘TBLFM’ keyword contains 含んでいます an odd number of dollar characters, this may cause problems 問題を起こすかもしれません with Font Lock in LATEX mode. As shown in the example you can fix this by adding 追加することによって an extra line inside the ‘comment’ environment that is used to balance the dollar expressions. If you are using AUCTEX with the font-latex library, a much better solution is to add the ‘comment’ environment to the variable 変数LaTeX-verbatim-environments を

```
% $ (optional extra dollar to keep Font Lock happy, see footnote)
\end{comment}
```

After editing, `C-c C-c` inserts 挿入します the translated table at the target location, between the two marker lines.

For hand-made custom tables, note 注意してください that the translator needs to skip the first two lines of the source table. Also また the command has to *splice* out the target table without the header and footer.

```
\begin{tabular}{lrrr}
Month & \multicolumn{1}{c}{Days} & Nr.\ sold & per day\\
% BEGIN RECEIVE ORGTBL salesfigures
% END RECEIVE ORGTBL salesfigures
\end{tabular}
%
\begin{comment}
#+ORGTBL: SEND salesfigures orgtbl-to-latex :splice t :skip 2
| Month | Days | Nr sold | per day |
|-------+------+---------+---------|
| Jan   |   23 |      55 |     2.4 |
| Feb   |   21 |      16 |     0.8 |
| March |   22 |     278 |    12.6 |
#+TBLFM: $4=$3/$2;%.1f
\end{comment}
```

The LaTeX translator function `orgtbl-to-latex` is already part of Orgtbl mode and uses a '`tabular`' environment to typeset the table and marks horizontal lines with `\hline`. For additional parameters to control output, see 節 A.5.3 [Translator functions], ページ 263:

'`:splice BOOLEAN`'
: When {{{var(BOOLEAN}}} が`nil` でない値のときには return only table body lines; i.e., すなわち not wrapped ラップされません in '`tabular`' environment. デフォルトは`nil` です。

'`:fmt FMT`' Format string to warp each field. It should contain 含むべきです'`%s`' を for the original field value. For example, 例えば to wrap each field value in dollar symbol, you could use '`:fmt "$%s$"`'. Format can also wrap a property list with column numbers and formats, for example 例えば'`:fmt (2 "$%s$" 4 "%s\\%%")`'. In place of a string, 文字列のかわりに、a function 関数を of one argument 1 つの引数を取る can be used; 使うことができます the function must return a formatted string.

'`:efmt EFMT`'
: Format numbers as exponentials. The spec should have 持つべきです'`%s`' を twice 2 回 for inserting mantissa and exponent, for example 例えば'`"%s\\times10^{%s}"`'. This これは may also be a property list with column numbers and formats, for example 例えば'`:efmt (2 "$%s\\times10^{%s}$" 4 "$%s\\cdot10^{%s}$")`'. After *EFMT* has been applied to a value, *FMT*---see above---is also applied. Functions with two arguments can be supplied instead かわりに of strings. 文字列の By default, デフォルトで no special formatting is applied. 適用されません

A.5.3 Translator functions

Orgtbl mode has built-in translator functions: `orgtbl-to-csv` (comma-separated values), `orgtbl-to-tsv` (TAB-separated values), `orgtbl-to-latex`, `orgtbl-to-html`, `orgtbl-to-texinfo`, `orgtbl-to-unicode` and `orgtbl-to-orgtbl`. They use the generic translator, `orgtbl-to-generic`, which これは delegates 委譲します translations 変換を to various export back-ends.

Properties passed to the function through the '`ORGTBL SEND`' line take precedence over properties defined inside the function. For example, 例えば this overrides the default LaTeX line endings, `\\`, with `\\[2mm]`:

```
#+ORGTBL: SEND test orgtbl-to-latex :lend " \\\\[2mm]"
```

For a new language translator, define a converter function. It can be a generic function, such as shown in this example. It marks a beginning and ending of a table with '`!BTBL!`' and '`!ETBL!`'; a beginning and ending of lines with '`!BL!`' and '`!EL!`'; and uses a TAB for a field separator:

```
(defun orgtbl-to-language (table params)
  "Convert the orgtbl-mode TABLE to language."
  (orgtbl-to-generic
   table
   (org-combine-plists
    '(:tstart "!BTBL!" :tend "!ETBL!" :lstart "!BL!" :lend "!EL!" :sep "\t")
    params)))
```

The documentation for the `orgtbl-to-generic` function shows 示しています a complete list of parameters, each of which can be passed through to `orgtbl-to-latex`, `orgtbl-to-texinfo`, and any other function using that generic function.

For complicated translations the generic translator function could be replaced by a custom translator function. Such a custom function must take two arguments and return a single string containing 含んでいる the formatted table. The first argument is the table whose lines are a list of fields or the symbol `hline`. The second argument is the property list consisting of parameters specified in the '`#+ORGTBL: SEND`' line. Please どうか share 共有してください your translator functions by posting them to the Org users mailing list, at `emacs-orgmode@gnu.org`.

A.6 Dynamic Blocks

Org supports サポートしています*dynamic blocks* in Org documents. They are inserted with begin and end markers like any other code block, but the contents are updated 更新されます automatically 自動的に by a user function.

You can insert 挿入することができます a dynamic block with `org-dynamic-block-insert-dblock`, which これは is bound バインドされています to *C-c C-x x* に by default. デフォルトで For example, 例えば*C-c C-x x c l o c k t a b l e RET* は inserts 挿入します a table that updates 更新する the work time（節 8.4 [Clocking Work Time], ページ 82, を見てください）。

Dynamic blocks can have names and function parameters. The syntax is similar to source code block specifications:

```
#+BEGIN: myblock :parameter1 value1 :parameter2 value2 ...
```

```
  ...
#+END:
```

These commands update dynamic blocks:

C-c C-x C-u (`org-dblock-update`)
: Update dynamic block at point.

C-u C-c C-x C-u
: Update all dynamic blocks in the current file.

Before updating a dynamic block, Org removes 取り除きます content between the '`BEGIN`' and '`END`' markers. Org then reads the parameters on the '`BEGIN`' line for passing to the writer function. If the function expects to access the removed content, then Org expects an extra parameter, '`:content`', on the '`BEGIN`' line.

The syntax for naming a writer function with a dynamic block labelled '`myblock`' is: `org-dblock-write:myblock`. Parameters come from the '`BEGIN`' line.

The following is an example of a dynamic block and a block writer function that updates the time when the function was last run:

```
#+BEGIN: block-update-time :format "on %m/%d/%Y at %H:%M"
  ...
#+END:
```

The dynamic block's writer function:

```
(defun org-dblock-write:block-update-time (params)
  (let ((fmt (or (plist-get params :format) "%d. %m. %Y")))
    (insert "Last block update at: "
            (format-time-string fmt))))
```

To keep dynamic blocks up-to-date in an Org file, use the function, `org-update-all-dblocks` in hook, such as `before-save-hook` といった The `org-update-all-dblocks` function does not run if the file is not in Org-mode.

Dynamic blocks, like any other block, can be narrowed with `org-narrow-to-block`.

A.7 Special Agenda Views

Org provides 提供しています a special hook to further limit items in agenda views: `agenda`, `agenda*`[3], `todo`, `alltodo`, `tags`, `tags-todo`, `tags-tree`. Specify a custom function that tests inclusion of every matched item in the view. This function can also skip as much as is needed.

For a global condition applicable to agenda views, use the 変数`org-agenda-skip-function-global` を Org uses a global condition with `org-agenda-skip-function` for custom searching.

This example この例は defines 定義します a function for a custom view showing TODO items with '`waiting`' status. Manually this is a multi-step search process, but with a custom view, this can be automated as follows:

[3] The `agenda*` view is the same as `agenda` except that it only considers *appointments*, i.e., すなわち scheduled and deadline items that have a time specification '`[h]h:mm`' in their time-stamps.

The custom function searches the subtree サブツリーを for the ‘`waiting`’ tag and returns `nil` on match. Otherwise そうでなければ it それは gives the location from where the search continues.

```
(defun my-skip-unless-waiting ()
  "Skip trees that are not waiting"
  (let ((subtree-end (save-excursion (org-end-of-subtree t))))
    (if (re-search-forward ":waiting:" subtree-end t)
        nil          ; tag found, do not skip
      subtree-end))) ; tag not found, continue after end of subtree
```

To use 使うには this custom function in a custom agenda command:

```
(org-add-agenda-custom-command
 '("b" todo "PROJECT"
   ((org-agenda-skip-function 'my-skip-unless-waiting)
    (org-agenda-overriding-header "Projects waiting for something: "))))
```

Note 注意してください that this also binds `org-agenda-overriding-header` to a more meaningful string suitable for the agenda view.

Search for entries with a limit set on levels for the custom search. This is a general approach to creating custom searches in Org. To include all levels, use ‘`LEVEL>0`’[4]. Then to selectively pick the matched entries, use `org-agenda-skip-function`, which これは also accepts 受け入れます Lisp forms, such as `org-agenda-skip-entry-if` や`org-agenda-skip-subtree-if` といった例です:

‘`(org-agenda-skip-entry-if 'scheduled)`’
: Skip スキップします current entry 現在のエントリーを if it has been scheduled.

‘`(org-agenda-skip-entry-if 'notscheduled)`’
: Skip スキップします current entry 現在のエントリーを if it has not been scheduled.

‘`(org-agenda-skip-entry-if 'deadline)`’
: Skip スキップします current entry 現在のエントリーを if it has a deadline.

‘`(org-agenda-skip-entry-if 'scheduled 'deadline)`’
: Skip スキップします current entry 現在のエントリーを if it has a deadline, or if it is scheduled.

‘`(org-agenda-skip-entry-if 'todo '("TODO" "WAITING"))`’
: Skip スキップします current entry 現在のエントリーを if the TODO keyword is TODO or WAITING.

‘`(org-agenda-skip-entry-if 'todo 'done)`’
: Skip スキップします current entry 現在のエントリーを if the TODO keyword marks a DONE state.

‘`(org-agenda-skip-entry-if 'timestamp)`’
: Skip スキップします current entry 現在のエントリーを if it has any timestamp, may also be deadline or scheduled.

[4] Note 注意してください that, for `org-odd-levels-only`, a level number corresponds to order in the hierarchy, not to the number of stars.

‘`(org-agenda-skip-entry-if 'regexp "regular expression")`’
: Skip スキップします current entry 現在のエントリーを if the regular expression matches in the entry.

‘`(org-agenda-skip-entry-if 'notregexp "regular expression")`’
: Skip スキップします current entry 現在のエントリーを unless the regular expression matches.

‘`(org-agenda-skip-subtree-if 'regexp "regular expression")`’
: Same as above, but check and skip the entire subtree.

The following is an example of a search for ‘`waiting`’ without the special function:

```
(org-add-agenda-custom-command
 '("b" todo "PROJECT"
   ((org-agenda-skip-function '(org-agenda-skip-subtree-if
                                'regexp ":waiting:"))
    (org-agenda-overriding-header "Projects waiting for something: "))))
```

A.8 Speeding Up Your Agendas

Some agenda commands slow down when the Org files grow in size or number. Here これらは are tips to speed up:

- Reduce the number of Org agenda files to avoid slowdowns due to hard drive accesses.
- Reduce the number of DONE and archived headlines so agenda operations that skip over these can finish faster.
- Do not dim blocked tasks:

  ```
  (setq org-agenda-dim-blocked-tasks nil)
  ```

- Stop preparing agenda buffers on startup:

  ```
  (setq org-agenda-inhibit-startup t)
  ```

- Disable tag inheritance for agendas:

  ```
  (setq org-agenda-use-tag-inheritance nil)
  ```

These options can be applied to selected agenda views. For more details about generation of agenda views, see 見てください the docstrings ドキュメント文字列を for the relevant variables, and this dedicated Worg page (`https://orgmode.org/worg/agenda-optimization.html`) for agenda optimization.

A.9 Extracting Agenda Information

Org Org-mode は provides 提供しています commands コマンドを to access agendas through Emacs batch mode. Through this command-line interface, agendas are automated for further processing or printing.

`org-batch-agenda` creates 作成します an agenda view アジェンダビューを in ASCII and outputs to standard output. This command takes one string parameter. When string consists of a single character, Org uses it as a key to `org-agenda-custom-commands`. These are the same ones available through the agenda dispatcher（節 10.2 [Agenda Dispatcher], ページ 108, を見てください）。

This example command line directly 直接 prints プリントします the TODO list to the printer:

```
emacs -batch -l ~/.emacs -eval '(org-batch-agenda "t")' | lpr
```

When the string parameter length is two or more characters, Org matches it with tags/TODO strings. For example, 例えば

this example command line prints items tagged with '`shop`', but excludes items tagged with '`NewYork`':

```
emacs -batch -l ~/.emacs                                      \
      -eval '(org-batch-agenda "+shop-NewYork")' | lpr
```

An example showing on-the-fly parameter modifications:

```
emacs -batch -l ~/.emacs                                      \
   -eval '(org-batch-agenda "a"                               \
           org-agenda-span (quote month)                      \
           org-agenda-include-diary nil                       \
           org-agenda-files (quote ("~/org/project.org")))'   \
   | lpr
```

which produces an agenda for the next 30 days from just the '`~/org/projects.org`' file.

For structured processing of agenda output, use `org-batch-agenda-csv` with the following fields:

category — The category of the item

head — The headline, without TODO keyword, TAGS and PRIORITY

type — The type of the agenda entry, can be

`todo`	selected in TODO match
`tagsmatch`	selected in tags match
`diary`	imported from diary
`deadline`	a deadline
`scheduled`	scheduled
`timestamp`	appointment, selected by timestamp
`closed`	entry was closed on date
`upcoming-deadline`	warning about nearing deadline
`past-scheduled`	forwarded scheduled item
`block`	entry has date block including date

todo — The TODO keyword, if any

tags — All tags including 含む inherited ones, separated by colons

date — The relevant date, like '`2007-2-14`'

time — The time, like '`15:00-16:50`'

extra — String with extra planning info

priority-l — The priority letter if any was given

priority-n — The computed numerical priority

If the selection of the agenda item was based on a timestamp, including 含む those items with 'DEADLINE' and 'SCHEDULED' keywords, then Org includes date and time in the output.

If the selection of the agenda item was based on a timestamp (or deadline/scheduled), then Org includes date and time in the output.

Here これは is an example 例です of a post-processing script in Perl での It takes the CSV output from Emacs and prints with a checkbox:

```
#!/usr/bin/perl

# define the Emacs command to run
$cmd = "emacs -batch -l ~/.emacs -eval '(org-batch-agenda-csv \"t\")'";

# run it and capture the output
$agenda = qx{$cmd 2>/dev/null};

# loop over all lines
foreach $line (split(/\n/,$agenda)) {
    # get the individual values
    ($category,$head,$type,$todo,$tags,$date,$time,$extra,
     $priority_l,$priority_n) = split(/,/,$line);
    # process and print
    print "[ ] $head\n";
}
```

A.10 Using the Property API

Here これは is a description 説明です。of the functions 機能の that can be used 使うことができる to work with properties. プロパティで作業するために

`org-entry-properties` **&optional** *pom which* [関数]
: Get all properties of the entry at point-or-marker *POM*. This includes the TODO keyword, the tags, time strings for deadline, scheduled, and clocking, and any additional properties defined in the entry. The return value is an alist. Keys may occur multiple times if the property key was used several times. *POM* may also be `nil`, in which case この場合には the current entry is used. If *WHICH* が`nil` か`all` の場合には get all properties. If *WHICH* が`special` か~standard~であれば、only get that subclass.

`org-entry-get` *pom property* **&optional** *inherit* [関数]
: Get value of *PROPERTY* for entry at point-or-marker *POM*. By default, デフォルトで this これあ only looks at properties defined locally in the entry. If *INHERIT* が`nil` でない値の場合には and the entry does not have the property, then also check higher levels of the hierarchy. If *INHERIT* is the symbol `selective`, use inheritance if and only if the setting of `org-use-property-inheritance` selects *PROPERTY* for inheritance.

`org-entry-delete` *pom property* [関数]
: Delete the property *PROPERTY* from entry at point-or-marker *POM*.

`org-entry-put` *pom property value* [関数]
Set *PROPERTY* to *VALUES* for entry at point-or-marker POM.

`org-buffer-property-keys` **&optional** *include-specials* [関数]
Get all property keys in the current buffer.

`org-insert-property-drawer` [関数]
Insert a property drawer for the current entry. Also

`org-entry-put-multivalued-property` *pom property* **&rest** *values* [関数]
Set *PROPERTY* at point-or-marker *POM* にある to *VALUES*. *VALUES* は should be a list of strings. 文字列のリストであるべきです They are concatenated, with spaces as separators.

`org-entry-get-multivalued-property` *pom property* [関数]
Treat the value of the property *PROPERTY* as a whitespace-separated list of values and return the values as a list of strings.

`org-entry-add-to-multivalued-property` *pom property value* [関数]
Treat the value of the property *PROPERTY* as a whitespace-separated list of values and make sure that *VALUE* is in this list.

`org-entry-remove-from-multivalued-property` *pom property value* [関数]
Treat the value of the property *PROPERTY* as a whitespace-separated list of values and make sure that *VALUE* is *not* in this list.

`org-entry-member-in-multivalued-property` *pom property value* [関数]
Treat the value of the property *PROPERTY* as a whitespace-separated list of values and check if *VALUE* is in this list.

`org-property-allowed-value-functions` [ユーザオプション]
Hook for functions supplying allowed values for a specific property. The functions must take a single argument, the name of the property, and return a flat list of allowed values. If '`:ETC`' is one of the values, use the values as completion help, but allow also other values to be entered. The functions must return `nil` if they are not responsible for this property.

A.11 Using the Mapping API

Org has sophisticated mapping capabilities to find all entries satisfying certain criteria. Internally, this functionality is used to produce 生成するために agenda views, アジェンダビューを but there is also an API もあります that can be used 使うことができます to execute 実行するために arbitrary functions 任意の関数を for each or selected entries. The main entry point for this API is:

`org-map-entries` *func* **&optional** *match scope* **&rest** *skip* [関数]
Call *FUNC* at each headline selected by *MATCH* in *SCOPE*.

FUNC is a function or a Lisp form. With point ポイントを positioned 置いて at the beginning of the headline, call the function without arguments. Org returns an alist of return values of calls to the function.

To avoid preserving point, Org wraps the call to *FUNC* in `save-excursion` form. After evaluation, Org moves point to the end of the line that was just processed. Search continues from that point forward. This これは may not always 常に work 動かないかもしれません as expected 期待どおりには under some conditions, such as if the current sub-tree was removed by a previous archiving operation. In such rare circumstances, Org skips the next entry entirely when it should not. To stop Org from such skips, make *FUNC* set the variable 変数`org-map-continue-from` を to a specific buffer position.

MATCH is a tags/property/TODO match. Org iterates only matched headlines. Org iterates over all headlines when *MATCH* is `nil` or `t`.

SCOPE determines the scope of this command. It can be any of:

`nil` The current buffer, respecting the restriction, if any.

`tree` The subtree started with the entry at point.

`region` The entries within the active region, if any.

`file` The current buffer, without restriction.

`file-with-archives`
: The current buffer, and any archives associated with it.

`agenda` All agenda files.

`agenda-with-archives`
: All agenda files with any archive files associated with them.

list of filenames
: If this is a list, all files in the list are scanned.

The remaining arguments are treated as settings for the scanner's skipping facilities. Valid arguments are:

`archive` Skip スキップします trees ツリーを with the '`ARCHIVE`' tag.

`comment` Skip スキップします trees ツリーを with the COMMENT keyword.

function or Lisp form
: Used as value for `org-agenda-skip-function`, so whenever いつでも the function returns `t` を*FUNC* は is called for that entry and search continues from the point where the function leaves it.

The mapping routine can call any arbitrary function, even functions that change meta data or query the property API（節 A.10 [Using the Property API], ページ 268, を見てください）。Here こららは are some handy functions:

`org-todo` **&optional** *arg* [関数]
: Change the TODO state of the entry. See the docstring of the functions for the many possible values for the argument *ARG*.

`org-priority` **&optional** *action* [関数]
: Change the priority of the entry. See the docstring of this function for the possible values for *ACTION*.

`org-toggle-tag` *tag* **&optional** *onoff* [関数]
: Toggle トグルします the tag *TAG* in the current entry. Setting *ONOFF* to either `on` or `off` does not toggle tag, but ensure that it is either on or off.

`org-promote` [関数]
: Promote the current entry.

`org-demote` [関数]
: Demote the current entry.

This example turns all entries tagged with '`TOMORROW`' into TODO entries with keyword '`UPCOMING`'. Org ignores entries in comment trees and archive trees.

```
(org-map-entries '(org-todo "UPCOMING")
                 "+TOMORROW" 'file 'archive 'comment)
```

The following example counts the number of entries with TODO keyword '`WAITING`', in all agenda files.

```
(length (org-map-entries t "/+WAITING" 'agenda))
```

付録 B History and Acknowledgments

B.1 From Carsten

Org was born in 2003, out of frustration over the user interface of the Emacs Outline mode. I was trying to organize my notes and projects, and using Emacs seemed to be the natural way to go. However, しかし having to remember eleven different commands with two or three keys per command, only to hide and show parts of the outline tree, that seemed entirely unacceptable to me. Also, また when using 使うときには outlines アウトラインを to take notes, ノートを取るために I constantly wanted to restructure the tree, organizing it parallel to my thoughts and plans. *Visibility cycling* and *structure editing* were originally implemented in the package '`outline-magic.el`', but quickly すぐに moved to the more general '`org.el`'. As this environment became comfortable for project planning, the next step was adding *TODO entries*, basic *timestamps*, and *table support*. These areas highlighted the two main goals that Org still has today: to be a new, outline-based, plain text mode with innovative and intuitive editing features, and to incorporate project planning functionality directly 直接 into a notes file.

Since the first release, literally thousands of emails to me or to the mailing list have provided a constant stream of bug reports, feedback, new ideas, and sometimes ときどき patches and add-on code. Many thanks to everyone who has helped to improve this package. I am trying to keep here a list of the people who had significant influence in shaping one or more aspects of Org. The list may not be complete, if I have forgotten someone, please どうか accept 受け入れて my apologies 私の謝罪を and let me know.

Before I get to this list, a few special mentions are in order:

Bastien Guerry
: Bastien has written a large number of extensions to Org (most of them integrated into the core by now), including 含む the LaTeX exporter and the plain list parser. His support during the early days was central to the success of this project. Bastien also invented Worg, helped establishing the Web presence of Org, and sponsored hosting costs for the orgmode.org website. Bastien stepped in as maintainer of Org between 2011 and 2013, at a time when I desperately needed a break.

Eric Schulte and Dan Davison
: Eric and Dan are jointly responsible for the Org Babel system, which これは turns Org into a multi-language environment for evaluating code and doing literate programming and reproducible research. This has become one of Org's killer features that define what Org is today.

John Wiegley
: John has contributed a number of great ideas and patches directly 直接 to Org, including 含む the attachment system ('`org-attach.el`'), integration with Apple Mail ('`org-mac-message.el`'), hierarchical dependencies of TODO items, habit tracking ('`org-habits.el`'), and encryption ('`org-crypt.el`'). Also, また the capture system is really an extended copy of his great '`remember.el`'.

Sebastian Rose
: Without Sebastian, the HTML/XHTML publishing of Org would be the pitiful work of an ignorant amateur. Sebastian has pushed this part of Org onto a much higher level. He also wrote '`org-info.js`', a Java script for displaying webpages derived from Org using an Info-like or a folding interface with single-key navigation.

See below for the full list of contributions! Again, please どうか let me 私に know 知らせてください what I am missing here!

B.2 From Bastien

I (Bastien) have been maintaining Org between 2011 and 2013. This appendix would not be complete without adding a few more acknowledgments and thanks.

I am first grateful to Carsten for his trust while handing me over the maintainership of Org. His unremitting support is what really helped me getting more confident over time, with both the community and the code.

When I took over maintainership, I knew I would have to make Org more collaborative than ever, as I would have to rely on people that are more knowledgeable than I am on many parts of the code. Here is これは a list リストです of the persons I could rely on, they should really 本当は be considered co-maintainers, either of the code or the community:

Eric Schulte
: Eric is maintaining the Babel parts of Org. His reactivity here kept me away from worrying about possible bugs here and let me focus on other parts.

Nicolas Goaziou
: Nicolas is maintaining the consistency of the deepest parts of Org. His work on '`org-element.el`' and '`ox.el`' has been outstanding, and it opened the doors for many new ideas and features. He rewrote many of the old exporters to use the new export engine, and helped with documenting this major change. More importantly (if that's possible), he has been more than reliable during all the work done for Org 8.0, and always 常に very reactive on the mailing list.

Achim Gratz
: Achim rewrote the building process of Org, turning some *ad hoc* tools into a flexible and conceptually clean process. He patiently coped with the many hiccups that such a change can create for users.

Nick Dokos
: The Org-mode mailing list would not be such a nice place without Nick, who patiently helped users so many times. It is impossible to overestimate such a great help, and the list would not be so active without him.

I received support from so many users that it is clearly impossible to be fair when shortlisting a few of them, but Org's history would not be complete if the ones above were not mentioned in this manual.

B.3 List of Contributions

- Russel Adams came up with the idea for drawers.
- Thomas Baumann wrote '`org-bbdb.el`' and '`org-mhe.el`'.
- Christophe Bataillon created the great unicorn logo that we use on the Org-mode website.
- Alex Bochannek provided a patch for rounding timestamps.
- Jan Böcker wrote '`org-docview.el`'.
- Brad Bozarth showed how to pull RSS feed data into Org files.
- Tom Breton wrote '`org-choose.el`'.
- Charles Cave's suggestion sparked the implementation of templates for Remember, which これらは are now 今は templates for capture.
- Pavel Chalmoviansky influenced the agenda treatment of items with specified time.
- Gregory Chernov patched support for Lisp forms into table calculations and improved XEmacs compatibility, in particular by porting '`nouline.el`' to XEmacs.
- Sacha Chua suggested copying some linking code from Planner.
- Baoqiu Cui contributed the DocBook exporter.
- Eddward DeVilla proposed and tested checkbox statistics. He also came up with the idea of properties, and that there should be an API for them.
- Nick Dokos tracked down several nasty bugs.
- Kees Dullemond used to edit projects lists directly 直接 in HTML and so inspired some of the early development, including 含む HTML export. He also asked for a way to narrow wide table columns.
- Thomas S. Dye contributed documentation on Worg and helped integrating the Org Babel documentation into the manual.
- Christian Egli converted the documentation into Texinfo format, inspired the agenda, patched CSS formatting into the HTML exporter, and wrote '`org-taskjuggler.el`'.
- David Emery provided a patch for custom CSS support in exported HTML agendas.
- Nic Ferrier contributed mailcap and XOXO support.
- Miguel A. Figueroa-Villanueva implemented hierarchical checkboxes.
- John Foerch figured out how to make incremental search show context around a match in a hidden outline tree.
- Raimar Finken wrote '`org-git-line.el`'.
- Mikael Fornius works as a mailing list moderator.
- Austin Frank works as a mailing list moderator.
- Eric Fraga drove the development of Beamer export with ideas and testing.
- Barry Gidden did proofreading the manual in preparation for the book publication through Network Theory Ltd.
- Niels Giesen had the idea to automatically 自動的に archive DONE trees.
- Nicolas Goaziou rewrote much of the plain list code.
- Kai Grossjohann pointed out key-binding conflicts with other packages.

- Brian Gough of Network Theory Ltd publishes the Org-mode manual as a book.
- Bernt Hansen has driven much of the support for auto-repeating tasks, task state change logging, and the clocktable. His clear explanations have been critical when we started to adopt the Git version control system.
- Manuel Hermenegildo has contributed various ideas, small fixes and patches.
- Phil Jackson wrote '`org-irc.el`'.
- Scott Jaderholm proposed footnotes, control over whitespace between folded entries, and column view for properties.
- Matt Jones wrote MobileOrg Android.
- Tokuya Kameshima wrote '`org-wl.el`' and '`org-mew.el`'.
- Shidai Liu ("Leo") asked for embedded LaTeX and tested it. He also provided frequent feedback and some patches.
- Matt Lundin has proposed last-row references for table formulas and named invisible anchors. He has also worked a lot on the FAQ.
- David Maus wrote '`org-atom.el`', maintains the issues file for Org, and is a prolific contributor on the mailing list with competent replies, small fixes and patches.
- Jason F. McBrayer suggested agenda export to CSV format.
- Max Mikhanosha came up with the idea of refiling.
- Dmitri Minaev sent a patch to set priority limits on a per-file basis. ファイルごとに
- Stefan Monnier provided a patch to keep the Emacs Lisp compiler happy.
- Richard Moreland wrote MobileOrg for the iPhone.
- Rick Moynihan proposed allowing multiple TODO sequences in a file and being able to quickly 素早く restrict the agenda to a subtree.
- Todd Neal provided patches for links to Info files and Elisp forms.
- Greg Newman refreshed the unicorn logo into its current form.
- Tim O'Callaghan suggested in-file links, search options for general file links, and tags.
- Osamu Okano wrote '`orgcard2ref.pl`', a Perl program to create a text version of the reference card.
- Takeshi Okano translated the manual and David O'Toole's tutorial into Japanese.
- Oliver Oppitz suggested multi-state TODO items.
- Scott Otterson sparked the introduction of descriptive text for links, among other things.
- Pete Phillips helped during the development of the TAGS feature, and provided frequent feedback.
- Martin Pohlack provided the code snippet to bundle character insertion into bundles of 20 for undo.
- T. V. Raman reported bugs and suggested improvements.
- Matthias Rempe (Oelde) provided ideas, Windows support, and quality control.
- Paul Rivier provided the basic implementation of named footnotes. He also acted as mailing list moderator for some time.
- Kevin Rogers contributed code to access VM files on remote hosts.

- Frank Ruell solved the mystery of the '`keymapp nil`' bug, a conflict with '`allout.el`'.
- Jason Riedy generalized the send-receive mechanism for Orgtbl tables with extensive patches.
- Philip Rooke created the Org reference card, provided lots of feedback, developed and applied standards to the Org documentation.
- Christian Schlauer proposed angular brackets around links, among other things.
- Paul Sexton wrote '`org-ctags.el`'.
- Tom Shannon's '`organizer-mode.el`' inspired linking to VM/BBDB/Gnus.
- Ilya Shlyakhter proposed the Archive Sibling, line numbering in literal examples, and remote highlighting for referenced code lines.
- Stathis Sideris wrote the '`ditaa.jar`' ASCII to PNG converter that is now packaged into Org's '`contrib/`' directory.
- Daniel Sinder came up with the idea of internal archiving by locking subtrees.
- Dale Smith proposed link abbreviations.
- James TD Smith has contributed a large number of patches for useful tweaks and features.
- Adam Spiers asked for global linking commands, inspired the link extension system, added support for Mairix, and proposed the mapping API.
- Ulf Stegemann created the table to translate special symbols to HTML, LaTeX, UTF-8, Latin-1 and ASCII.
- Andy Stewart contributed code to '`org-w3m.el`', to copy HTML content with links transformation to Org syntax.
- David O'Toole wrote '`org-publish.el`' and drafted the manual chapter about publishing.
- Jambunathan K. contributed the ODT exporter.
- Sebastien Vauban reported many issues with LaTeX and Beamer export and enabled source code highlighting in Gnus.
- Stefan Vollmar organized a video-recorded talk at the Max-Planck-Institute for Neurology. He also inspired the creation of a concept index for HTML export.
- Jürgen Vollmer contributed code generating the table of contents in HTML output.
- Samuel Wales has provided important feedback and bug reports.
- Chris Wallace provided a patch implementing the '`QUOTE`' block.
- David Wainberg suggested archiving, and improvements to the linking system.
- Carsten Wimmer suggested some changes and helped fix a bug in linking to Gnus.
- Roland Winkler requested additional key bindings to make Org work on a TTY.
- Piotr Zielinski wrote '`org-mouse.el`', proposed agenda blocks and contributed various ideas and code snippets.
- Marco Wahl wrote '`org-eww.el`'.

付録 C GNU Free Documentation License

Version 1.3, 3 November 2008

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of ‘‘copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The ‘‘Document”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as ‘‘you”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A ‘‘Modified Version” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A ‘‘Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The ‘‘Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The ‘‘Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A ‘‘Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not ‘‘Transparent” is called ‘‘Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The ‘‘Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, ‘‘Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

The ‘‘publisher” means any person or entity that distributes copies of the Document to the public.

A section ‘‘Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as ‘‘Acknowledgements”, ‘‘Dedications”, ‘‘Endorsements”, or ‘‘History”.) To ‘‘Preserve the Title” of such a section when you modify the Document means that it remains a section ‘‘Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

   You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

   You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

   If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

   If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

   If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

   It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

   You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
H. Include an unaltered copy of this License.
I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at

your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled ‘‘Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties---for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

   You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

   The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

   In the combination, you must combine any sections Entitled ‘‘History” in the various original documents, forming one section Entitled ‘‘History”; likewise combine any sections Entitled ‘‘Acknowledgements”, and any sections Entitled ‘‘Dedications”. You must delete all sections Entitled ‘‘Endorsements.”

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an ‘‘aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

   If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled ‘‘Acknowledgements”, ‘‘Dedications”, or ‘‘History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

   However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

   Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

   Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

    The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See `http://www.gnu.org/copyleft/`.

    Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

    "Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

    "CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

    "Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

    An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

    The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.  A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with. . . Texts." line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

16 Main Index

（インデックスがありません）

17 Key Index

（インデックスがありません）

18 Command and Function Index

（インデックスがありません）

19 Variable Index

This is not a complete index of variables and faces, only the ones that are mentioned in the manual. For a more complete list, use *M-x org-customize* and then click yourself through the tree.

（インデックスがありません）