

Doom Emacs / Evil / Treemacs — Cheatsheet (только хоткеи)

Почти всё ниже — в **normal mode** (`Esc` сначала).

Режимы

- `Esc` — в normal mode (выход из вставки)
 - `i` — вставка перед курсором
 - `a` — вставка после курсора
 - `v` — visual-режим (выделение)
-

Файл / буферы / табы

Файлы

- `C-x C-f` — открыть файл (ввести путь → `Enter`)
- `:w` — сохранить файл
- `C-x C-s` — сохранить файл
- `SPC f s` — сохранить файл
- `SPC b d` — закрыть текущий файл (kill buffer)
- `:bd` — закрыть текущий файл (buffer delete)
- `C-x k → Enter` — закрыть текущий файл

Буферы (открытые файлы)

- `SPC b b` — список буферов → выбрать → `Enter`
- `SPC b n` — следующий буфер
- `SPC b p` — предыдущий буфер
- `C-^` (`Ctrl+6`) — переключиться на предыдущий буфер (туда-сюда)

Табы

- `g t` — следующий таб
 - `g T` — предыдущий таб
-

Undo / Redo

- `u` — undo
 - `C-r` — redo
-

Навигация по файлу (vim-движок)

- `h / j / k / l` — влево / вниз / вверх / вправо
- `w / b` — вперёд / назад по словам
- `0` — начало строки
- `$` — конец строки
- `gg` — начало файла
- `G` — конец файла
- `C-d` — полэкрана вниз
- `C-u` — полэкрана вверх
- `C-f` — экран вниз
- `C-b` — экран вверх

Поиск и подсветка

Внутри файла

- `/текст → Enter` — поиск вперёд
- `n / N` — следующее / предыдущее совпадение
- `*` — найти слово под курсором (вперёд)
- `#` — найти слово под курсором (назад)
- `g*` — найти подстроку под курсором (вперёд)
- `:noh → Enter` — убрать подсветку поиска

По директории / проекту

- `SPC s d` — поиск по **текущей директории**
 - `SPC /` — поиск по **проекту**
-

Go / LSP навигация

(курсор на идентификаторе, normal mode)

- `SPC s i` — список функций/символов в файле (imenu) → выбрать → `Enter`
 - `gd` — перейти к определению
 - `C-o` — назад по истории переходов
 - `K` — показать документацию / информацию под курсором
 - `] g` — следующая ошибка/варнинг
 - `[g` — предыдущая ошибка/варнинг
-

Сплиты (окна Emacs)

Создание

- `C-w v` — вертикальный сплит (лево/право)
- `C-w s` — горизонтальный сплит (верх/низ)
- `C-x 3` — вертикальный сплит
- `C-x 2` — горизонтальный сплит

Переключение между окнами

- `C-w h` — влево
- `C-w l` — вправо
- `C-w k` — вверх
- `C-w j` — вниз
- `C-w w` — следующее окно по кругу
- `C-x o` — следующее окно по кругу

Закрытие / выравнивание

- `C-w c` — закрыть текущее окно
 - `C-x 0` — закрыть текущее окно
 - `C-w o` — оставить только текущее окно
 - `C-x 1` — оставить только текущее окно
 - `C-w =` — выровнять размеры всех окон
-

Treemacs

- **SPC o p** — показать / спрятать Treemacs
 - В Treemacs:
 - **j / k** — вверх / вниз по дереву
 - **RET** — открыть файл
 - **o** — открыть файл (аналогично)
 - Переключиться в окно Treemacs:
 - **SPC b b** → выбрать буфер *Treemacs* → Enter
 - Вернуться в файл:
 - **C-x o / C-w h/j/k/l** или открыть файл из дерева (RET)
-

vterm

- **M-x vterm** → Enter — открыть vterm
 - **exit** → Enter — закрыть shell внутри vterm
 - **C-x k** → Enter — закрыть буфер vterm
-

Перенос строк

- **M-x visual-line-mode** — включить/выключить перенос строк по словам в текущем буфере
-

Копирование / вставка (внутри Emacs)

В normal mode:

- **yy** — копировать текущую строку
- **3yy** — копировать 3 строки (пример)
- **v** → выделить → **y** — копировать выделение
- **p** — вставить после
- **P** — вставить перед

Emacs / Doom / Evil Hotkey Cheatsheet (из текущей сессии)

1. Поиск (ripgrep, Doom, Embark/Consult)

Базовый поиск по директории

- **SPC s d** — поиск по директории (Doom, consult-ripgrep).
- В минибуфере после ввода запроса:
 - **C-c C-e** — экспорт результатов в обычный буфер (*Embark Collect*), где:
 - * **SPC t l** — включить перенос строк (toggle-truncate-lines).
 - * **zL / zH** — горизонтальный скролл вправо/влево (evil).
 - * **RET** — перейти к найденному месту.
 - * **q** — закрыть буфер.

Поиск по длинным строкам логов

Кастомный биндинг (который мы обсуждали) для поиска по логам и .gz:

```
(defun my/rg-gz-logs (dir pattern)
  "Быстрый поиск по *.log и *.log.gz в DIR с помощью ripgrep -z."
  (interactive
   (list
```

```

(read-directory-name "Search in directory: " default-directory)
(read-from-minibuffer "rg -z pattern: " (thing-at-point 'symbol)))
(let ((default-directory dir))
  (compilation-start
    (format "rg -z -n --no-heading --color never -g'*.\log' -g'*.\log.gz' %s ."
           (shell-quote-argument pattern))
    'grep-mode
    (lambda (_)
      "*rg-gz*")))

(map! :leader
  "s D" #'my/rg-gz-logs)

```

После этого:

- **SPC s D** — быстрый поиск по логам (*.log, *.log.gz) в выбранной директории.
- C-c C-k в буфере *rg-gz* /*grep* — остановить бегущий поиск (kill-compilation).

Фильтрация результатов по содержимому

В любом grep-подобном буфере (включая *Embark Collect* / *rg-gz*):

- **:%g/ТЕКСТ/d** — удалить все строки, содержащие ТЕКСТ.
 - Пример: оставить строки с key_tag:, но без key_tag:222002:
:%g/key_tag:222002/d
-

2. Magit и Git

Статус и ветки

- **C-x g** — magit-status (главный вход в Magit).
- В статусе:
 - **b b** — выбрать / переключить ветку (checkout).
 - **q** — закрыть Magit-буфер.

Blame / история строки

- **SPC g b** — magit-blame-addition:
 - показывает blame по строкам текущего файла.
 - RET по строке — открыть коммит, где она менялась.
 - **q** — выйти из просмотра коммита или снять blame.

Echo-blame (минималистично)

- **M-x magit-blame-echo** — при перемещении по строкам пишет blame в эхо-области (внизу).
 - Можно повесить на хоткей, например:

```

(map! :leader
  "g B" #'magit-blame-echo)

```

3. Blamer (оверлей коммита прямо на строке)

Если установлен blamer.el:

Пример конфига:

```
(use-package! blamer
  :custom
  (blamer-idle-time 0.2)
  (blamer-min-offset 0)
  (blamer-view 'overlay)
  :config
  (set-face-attribute 'blamer-face nil
    :foreground "#7a88cf"
    :background nil
    :italic t)
  (global-blamer-mode 1))
```

Поведение:

- Останавливаешь курсор на закоммиченной строке → справа появляется подпись вида: user | дата | часть commit message.
-

4. Evil: блоки, вставка в несколько строк, сдвиг

Вставить/дописать текст в КОНЕЦ нескольких строк

1. Встать на первую строку.
2. **C-v** — visual block.
3. j/k — выделить нужное количество строк.
4. \$ — растянуть блок до конца строк.
5. **Атекст ESC** — дописать текст в конец каждой строки.

Вставить префикс в НАЧАЛО нескольких строк

1. Встать в начало первой строки (0 или ^).
2. C-v — visual block, j/k — вниз по строкам.
3. **Итекст ESC** — вставить текст в начало каждой строки.

Вставка по фиксированной колонке

1. Встать в нужную колонку на первой строке.
2. C-v → j/k — выделить строки.
3. I... ESC или A... ESC — вставка/дописка в одной колонке для всех строк.

Сдвиг блока (indent / unindent)

В visual или visual block режиме:

- > — сдвиг блока вправо (по shiftwidth/tab).
- < — сдвиг блока влево.

Доп. фишки:

- gv — повторно выделить последний visual регион (удобно: > gv > gv).

Если хочется двигать TAB'ом:

```
(map! :v "<tab>" #'evil-shift-right
      :v "<backtab>" #'evil-shift-left) ;; Shift+Tab
```

5. Go + LSP (gopls)

Подсказки / completion после точки

- Пишешь `cfg.` и жмёшь:
 - **C-SPC** — `completion-at-point` → LSP выдаёт методы/поля для `cfg.`

Чтобы completion вылезал сам:

- Для corfu:

```
(after! corfu
  (setq corfu-auto t
        corfu-auto-prefix 1
        corfu-auto-delay 0.0))
```

- Для company:

```
(after! company
  (setq company-idle-delay 0.2
        company-minimum-prefix-length 1))
```

Найти все использование функции / символа

Курсор на имени функции/переменной:

- **M-?** — `xref-find-references`:
 - открывает буфер `*xref*` со всеми вхождениями.
 - `n / p` — ходить по результатам.
 - `RET` — прыгнуть в код.
 - `q` — закрыть.

Альтернатива:

- `M-x lsp-find-references` — тот же эффект, но напрямую через LSP.

Автодобавление импортов (code actions)

Курсор на неразрешённом идентификаторе (`undeclared name`, не хватает импорта):

- **M-RET** — меню LSP code actions (в Doom может быть SPC с `l a` / SPC с `a`).
- В списке выбрать:
 - `Add import "xxx"` или
 - `Add all missing imports`.

Форматирование + импорты при сохранении

```
(after! go-mode
  (add-hook 'go-mode-local-vars-hook
    (lambda ()
      (add-hook 'before-save-hook #'lsp-format-buffer nil t)
      (add-hook 'before-save-hook #'lsp-organize-imports nil t)))
```

6. Go: запуск тестов

В Go-файле (Doom, go-mode):

- Тесты текущего файла:

```
SPC m t f
```

(go-test-current-file)

- Тест под курсором (func TestXxx):

```
SPC m t t
```

(go-test-current-test)

- Тесты всего пакета (директории):

```
SPC m t p
```

Результаты — в *go-test* / *compilation*:

- n / p — по ошибкам/варнингам.
- RET — перейти в соответствующую строку кода.
- q — закрыть буфер.