17 приёмов в Клод Код

Из трёх месяцев ежедневной работы. Не теория - практика

Что вы получите из этого гайда

- 17 практических приёмов, которые экономят часы работы каждую неделю.
- 4 блока с понятной логикой: общение → контекст → настройка → масштабирование.
- 3 топ-приёма для срочного внедрения - если попробуете только их, уже почувствуете разницу.
- Готовые правила для `CLAUDE.md`, которые можно скопировать в свой файл.

Время чтения: ≈ 12 минут. **Уровень:** для тех, кто уже неделю работает с Claude Code и хочет выжать из инструмента больше.

> Если ещё не пользовались Claude Code - сначала возьмите «Клод Код за вечер», там все базовые шаги. Потом возвращайтесь сюда за приёмами.

С чего всё началось

Когда я только начала работать с Claude Code, я тратила кучу времени на вещи, которые можно было сделать в два раза быстрее. Просто потому что не знала маленьких хитростей.

Эти 17 приёмов - то, что я собрала за три месяца ежедневной работы. Каждый из них экономит несколько минут или часов - в зависимости от того, как часто вы попадаете в нужную ситуацию. Не нужно внедрять все 17 за один день. Возьмите 2-3 которые попадают в ваши боли - этого уже хватит, чтобы работа пошла быстрее.

Дополнительно я записала видео - там показано всё в реальном времени с примерами на экране.

Блок 1. Общение с Claude эффективнее

Приём 1. Попросите Claude задать вам вопросы

Часто бывает так: вы что-то хотите сделать, но не понимаете, как сформулировать задачу. Вместо того чтобы мучиться с формулировкой - переверните ситуацию.

Скажите Claude: *«Я хочу автоматизировать обработку входящих писем. Задай мне вопросы, чтобы понять задачу лучше».* Claude начнёт спрашивать: какие письма, на какие категории делить, что с ними делать, где хранить, есть ли исключения.

Вы отвечаете - и в итоге получаете точную постановку задачи, которую с первого раза сами бы не сформулировали так чётко. Чем сложнее процесс - тем сильнее эффект от этого приёма.

Приём 2. Очередь сообщений

Многие не знают, что Claude Code поддерживает очередь сообщений. Вы отправляете запрос, он его обрабатывает, и пока обрабатывает - можно набирать следующее сообщение. И второе, и третье. Claude по очереди их выполнит.

Удобно, когда вы точно знаете, что хотите попросить дальше. Не нужно сидеть и ждать пока закончится текущая задача.

Например: *«Привет. Давай продолжим скилл по обработке лидов»* - и пока Claude ищет вчерашний контекст, я сразу пишу: *«Добавь Google Sheets куда будем сохранять данные»*, а следом: *«И покажи список источников, которые мы анализировали вчера»*. Claude обработает их по порядку.

Приём 3. Голосовой ввод

Мы говорим в 3-4 раза быстрее, чем печатаем. Эту разницу нужно использовать.

Я пользуюсь приложением WisprFlow - оно хорошо распознаёт смешанную речь (русский + английские термины), а такие фразы у меня в запросах постоянно.

Особенно полезно для описания задач: проще объяснить голосом, как будто рассказываете задачу новому коллеге, чем формулировать письменно.

Приём 4. Дайте критерии проверки в самом запросе

Это самый мощный приём по эффективности промптов. В конец каждого запроса добавляйте, как Claude может проверить результат сам.

Слабый запрос: «Создай скрипт для обработки файлов». **Сильный запрос:** «Создай скрипт для обработки файлов. Запусти его на тестовой папке. Если найдёшь ошибки - исправь и проверь снова».

Когда Claude понимает, как проверить свою работу - качество вырастает драматически. Он сам находит ошибки, сам исправляет, отдаёт уже проверенный результат. Возьмите за привычку: каждая задача заканчивается словами «проверь результат».

Блок 2. Управление контекстом

Приём 5. Новый разговор на каждую задачу

Когда в одном разговоре вы делаете разные задачи - обработали счета, потом написали письмо, потом собрали отчёт - Claude держит в голове всё одновременно. Чем больше накапливается, тем хуже он работает.

Простое правило: новая задача - новый разговор. Это занимает секунду, а качество вырастает заметно.

Если не хочется закрывать окно - используйте команду `/clear`. Она очищает контекст прямо в текущем разговоре. Удобно когда задача закончена и хочется начать следующую с чистого листа.

Исключение: если задачи связаны и предыдущий контекст важен - оставайтесь в одном разговоре.

Приём 6. `/compact` с фокусом

Когда вы много работаете в одном окне, появляется индикатор использованного контекста (как правило при 70-80% лимита). При 100% Claude автоматически сжимает всё, что было в окне.

Но вы можете вызвать `/compact` сами на любом этапе. Главная фишка: при вызове можно уточнить, какие детали Claude точно должен сохранить, а остальное сжать.

Это страховка от потери важного контекста при автоматическом сжатии.

Приём 7. Escape и Rewind

Escape - останавливает Claude в любой момент. Попросили запустить скилл, передумали - нажали Escape, видите «Interrupted». Всё, работа над текущей задачей прекращена.

Rewind - откатывает разговор и файлы к любой предыдущей точке. Claude сделал изменения в файлах, результат не нравится - откатываете к нужному сообщению, изменения в файлах тоже откатываются. Как «Undo» в Word, только для всего проекта сразу.

Это даёт свободу экспериментировать: всегда можно вернуться к рабочей версии, не боясь сломать.

Приём 8. Возвращение к прошлым разговорам

Закрыли VS Code, ушли на обед, вернулись - и думаете, что разговор потерян. Нет. Claude Code хранит всю историю.

В VS Code достаточно кликнуть на значок Session History в боковой панели - и увидите все разговоры за последние дни. Открываете нужный, продолжаете с того места, где остановились.

Блок 3. Настройка Claude под себя

Все приёмы из этого блока - правила, которые добавляются в файл `CLAUDE.md`. Это инструкция, которую Claude читает перед каждой работой. Если не знаете, что такое `CLAUDE.md` - см. «Клод Код за вечер», там разобрано.

Приём 9. Правило «Оспаривай мои решения»

Claude по умолчанию ведёт себя как вежливый подчинённый - со многим соглашается. Но нам нужен честный советник.

Добавьте в `CLAUDE.md`:

> *«Если мой подход не оптимальный или есть лучшее решение - скажи прямо. Не соглашайся просто потому, что я так сказала».*

После этого Claude начинает говорить вещи вроде: *«Вы предложили сделать через таблицу, но для вашей задачи лучше база данных. Вот почему...»*. Одно правило - и качество решений вырастает.

Приём 10. Правило «Оценивай качество по шкале 1-10»

Добавьте в `CLAUDE.md`:

> *«Когда создаёшь текст, отчёт, любой контент - оцени результат по шкале от 1 до 10. Если ниже 8 - доработай сам, не показывай мне».*

Claude становится самокритичным. Вместо «готово» вы получаете результат, прошедший внутренний контроль качества.

Приём 11. Правило «Структура ответа»

Когда Claude работает над сложной задачей, иногда непонятно, что от вас требуется дальше. Решается одним правилом в `CLAUDE.md`:

> *«Каждый ответ давай по формату: что сделано, что нужно от меня (если нужно), следующие шаги».*

Каждый ответ становится структурированным. Видно, где он закончил, что от вас ожидается, и куда мы движемся.

Приём 12. Memory vs CLAUDE.md - в чём разница

Многие путают эти два механизма, а они работают по-разному.

`CLAUDE.md` - правила, которые вы пишете вручную. Например: «отвечай кратко», «используй такой-то стиль», «папки оформляй так-то». Файл статичный, сам не меняется.

Memory - это то, что Claude запоминает автоматически в процессе работы. Не правила, а **факты о вас**: кто вы, чем занимаетесь, какие предпочтения, над чем работаете, какие скиллы создавали. В новом разговоре Claude уже знает, кто вы - не нужно представляться заново.

Полезно периодически открывать файл памяти и смотреть, что там накопилось. Когда сообщаете Claude что-то важное - можно сказать: *«Запомни это, чтобы больше к этому не возвращаться»*. Например: *«Никогда не используй длинное тире, только короткое»* - идёт в память, и больше не нужно об этом напоминать.

Приём 13. Правило «Предлагай улучшения»

Добавьте в `CLAUDE.md`:

> *«После каждой выполненной задачи коротко предложи, что можно оптимизировать, улучшить или автоматизировать в этом процессе».*

Claude начинает работать как консультант: вы решаете текущую задачу, а он попутно показывает возможности, о которых вы не задумывались.

У меня после этого правила Claude иногда говорит: *«Вы делаете эту задачу каждую неделю - хотите, я создам скилл, который автоматизирует процесс?»*. У всех нас есть слепые зоны - этот приём их подсвечивает.

Блок 4. Масштабирование работы

Приём 14. Несколько вкладок параллельно

Не обязательно работать в одном окне. Можно открыть несколько вкладок VS Code, и в каждой - своя задача в Claude Code.

В одной вкладке Claude собирает скилл по обработке лидов, во второй параллельно делает research по маркетинговым агентствам в Минске. Оба работают одновременно, не мешая друг другу. Скорость работы вырастает в разы.

Приём 15. Subagents для больших задач

Это уже для тех, кто уверенно работает с Claude Code. Если задача большая - попросите Claude разбить её на части и делегировать сабагентам.

Пример: *«Создай 3 субагентов. Первый соберёт данные из писем за март. Второй подготовит шаблон отчёта. Третий проверит результаты первых двух».*

Сабагенты работают параллельно, каждый фокусируется на одной задаче, не отвлекаясь на остальное. Результат быстрее и качественнее.

Приём 16. Skills - автоматизация повторяющихся задач

Если вы делаете что-то каждую неделю - превратите это в скилл. Скилл - это набор инструкций, который выполняется по одной команде. Настраиваете один раз, потом запускаете снова и снова.

У меня, например, скилл анализирует мейлы Gmail за заданный период и раскладывает их на три группы: требуют моего ответа, средний приоритет, неважные (помечает прочитанными автоматически). Превращает рутину чтения почты в 30 секунд работы.

Любой повторяющийся процесс - кандидат на скилл. Подробнее в материале «Скиллы в Клод Код с нуля».

Приём 17. MCP-серверы для подключения внешних сервисов

MCP (Model Context Protocol) - стандарт подключения внешних инструментов: Google Sheets, Notion, базы данных, мессенджеры.

Без MCP Claude работает только с файлами на вашем компьютере. С MCP он может зайти в Airtable, заполнить таблицу, выгрузить данные, проверить информацию во внешней системе.

Подключение MCP-сервера обычно требует терминал (не VS Code) и API-ключ от целевого сервиса. Каждая команда подключения зависит от сервера. Это уже следующий уровень - в одном из следующих гайдов разберём подробнее.

Если возьмёте только три приёма

17 - это много. Если хочется внедрить только то, что даст эффект быстрее всего, начните с этих:

1. **Приём 4. Критерии проверки в запросе** - одна фраза в конце промпта поднимает качество результата кратно.
2. **Приём 5. Новый разговор на каждую задачу** - свежий контекст всегда лучше работает, чем перегруженный.
3. **Приёмы 9-13. Четыре правила в `CLAUDE.md`** - оспаривай решения, оценивай по шкале, структурируй ответ, предлагай улучшения. Один раз настроили - работает на каждой задаче.

Остальные осваивайте постепенно, под свои сценарии.

После этого гайда

Естественное продолжение в каталоге AIAtlas:

- **«98% токенов впустую»** - 16 советов как не сжигать контекст и продлить полезное окно. Углубление в Блок 2 этого гайда.
- **«Скиллы в Клод Код с нуля»** - как собрать первый скилл за вечер. Развитие приёма 16.
- **«Клод Код + Notebook LM»** - связка двух инструментов для исследований и контента.

Подписывайтесь, чтобы видеть новое

YouTube · **@ai_with_marina** Подробные разборы AI-инструментов и пошаговые гайды для русскоязычной аудитории. Новое видео раз в неделю.

Telegram · **@ai_with_marina** Короткие заметки, анонсы новых материалов AIAtlas, обсуждения и быстрые ответы на вопросы.

Если что-то в гайде не работает или нашли неточность - напишите в Telegram-канал, отвечу.