Промпт-шорткаты для Claude: официальные приёмы вместо самодеятельности

Списков «32 хака для Claude» в сети много, но большинство без источника и без проверки. Ниже — приёмы промптинга, которые реально описаны в официальной документации Anthropic по prompt engineering, а не собраны из случайных постов.

> **Важно:** когда extended thinking выключено, Claude Opus 4.5 болезненно реагирует именно на слово «think» и его формы — в такой ситуации замени его на «рассмотри», «оцени» или «разбери шаг за шагом».

1. Ясность и прямота — золотое правило

- Claude хорошо реагирует на чёткие, явные инструкции; если нужно поведение «сверх ожидаемого» — прямо попроси об этом, а не рассчитывай, что модель сама это домыслит из расплывчатого промпта.
- **Правило:** покажи свой промпт коллеге, у которого минимум контекста по задаче, и попроси выполнить по нему — если он запутается, запутается и Claude.
- Будь конкретен насчёт нужного формата и ограничений вывода; если порядок шагов важен — давай инструкции пронумерованным списком.

2. Роль в системном промпте

- Роль, заданная в системном промпте, фокусирует поведение и тон Claude под конкретную задачу — разницу даёт уже одно предложение.
- Пример: `system="You are a helpful coding assistant specializing in Python."` меняет глубину и стиль ответа заметнее, чем кажется на первый взгляд.

3. Пошаговое рассуждение без extended thinking

- Когда встроенное расширенное мышление выключено, ручной chain-of-thought всё равно работает: попроси Claude рассуждать шаг за шагом и раздели рассуждение и финальный ответ тегами вида `<thinking>` и `<answer>`.
- **Самопроверка:** фраза «прежде чем закончить, проверь свой ответ по [критериям]» надёжно ловит ошибки — особенно в коде и математике.

4. Формат вместо запретов

- Говори Claude, что делать, а не что НЕ делать: вместо «не используй markdown» — «ответ должен состоять из плавно связанных абзацев прозы».
- XML-теги работают как явный указатель формата: можно попросить писать текст внутри тега вроде `<smoothly_flowing_prose_paragraphs>`, чтобы обозначить нужную структуру без описания словами.
- Стиль самого промпта влияет на стиль ответа — если markdown всё равно просачивается в вывод, убери markdown и из своего промпта тоже.

5. Явные инструкции для действий и инструментов

- Модель следует инструкциям буквально: на «можешь предложить изменения» она часто просто предлагает варианты вместо того, чтобы сразу их внести — если нужно действие, попроси об этом прямо.
- Чтобы Claude был проактивнее по умолчанию, в системный промпт добавляется блок вроде `<default_to_action>` с инструкцией внедрять изменения, а не только предлагать их.

- Для обратного эффекта — более осторожного поведения — используется блок `<do_not_act_before_instructions>`, чтобы Claude действовал только по явному запросу.

6. Примеры и XML-структура вместо длинных объяснений

- Несколько подобранных примеров (few-shot) надёжнее задают формат, тон и структуру ответа, чем словесное описание, — оптимально 3–5 примеров, релевантных и разных между собой.
- Примеры оборачиваются в тег `<example>` (несколько сразу — в `<examples>`), чтобы Claude отличал их от остальных инструкций.
- XML-теги в целом помогают разделить инструкции, контекст, примеры и вводные данные без путаницы — используй одинаковые понятные имена тегов по всему промпту.