ЧАСТЬ 8 · ИИ В РАБОТЕ

ИТ-Коктель с Анной Фединой

Новый контекст. Сайт за 30 минут.

Кейс Kimi → Codex: как пересобрать сайт из накопленных решений



Опыт · Шаблоны · Практика / Сентябрь 2026

СЛОВО АВТОРА

Сайт не пришлось объяснять с нуля

Первую версию я собрала в Kimi. После изменения контекста проекта Codex пересобрал сайт за 30 минут.



Анна Федина
CIO · Автор канала

У этого кейса две части. Сначала появился сайт, собранный в Kimi. Затем изменился контекст проекта: прежняя версия уже не соответствовала тому, что требовалось дальше. Codex пересобрал сайт из обновлённого контекста за 30 минут.

Для меня здесь важен не забег двух инструментов. Быстрая пересборка стала возможной в ситуации, когда проект уже имел накопленный контекст. Не пришлось начинать разговор с пустой страницы и заново объяснять всю задачу.

Что показывает этот опыт

Тридцать минут - время конкретной пересборки, которое я привожу из своего опыта. Оно не включает всю предшествующую работу над проектом и не является обещанием собрать любой сайт за полчаса.

> Реальная последовательность
> Первая версия в Kimi → изменение контекста → передача обновлённой задачи в Codex → пересобранный сайт.
> Публичные показатели сайта и доставка заявок в этот вывод не включены.

Дальше я разложу кейс в повторяемую методику. Файлы, контракт формы и проверки в примерах ниже учебные: они объясняют перенос, но не описывают неподтверждённый стек моего сайта.

> **Результат выпуска: пакет передачи между инструментами, карта изменений сайта и приёмка новой версии. Сохранённый контекст превращается в рабочий актив.**

ВЫПУСК 8 · ШАГ 1

Сначала определить, что изменилось

Нельзя поручить пересборку словом «обнови», если старые и новые решения лежат вперемешку.

> Учебная карта изменений
> Было: страница объясняет отдельную демонстрацию.
> Стало: страница представляет несколько направлений проекта.
> Сохранить: подтверждённые факты, контакты, фирменный стиль.
> Пересмотреть: структуру, первый экран и главный призыв.
> Исключить: отменённые обещания и старые условия.

Эта карта - пример оформления, не фактический список изменений моего сайта. В своей задаче подставьте реальные решения. Полезно указывать причину: иначе исполнитель воспримет устаревший абзац как текст, который обязательно нужно сохранить.

Промпт: собрать изменения из контекста

> Прочитай утверждённый бриф, актуальную карточку проекта и предыдущую структуру сайта. Составь таблицу: старое решение / новое решение / основание / затронутый блок / проверка. Не используй отменённый факт только потому, что он есть в старом макете. Если источники противоречат друг другу, покажи точное расхождение.

Ожидаю короткий список изменений с опорой на документы. Пересборка всего сайта не означает, что надо переписать каждое предложение. Согласованные факты и узнаваемые элементы можно сохранить.

Граница первого прохода

В первом проходе фиксирую содержание и сценарий пользователя. Новая интеграция, платная инфраструктура или другой получатель заявок не должны возникать автоматически из просьбы изменить внешний вид.

> **Для быстрой работы нужен не максимальный объём контекста, а актуальные решения и понятная граница изменения.**

ВЫПУСК 8 · ШАГ 2

Передача Kimi → Codex

Передаю результат и основания. Переписка целиком редко является лучшим техническим заданием.

```
site-handoff/
  brief-current.md
  changes.md
  content-approved.md
  assets/
  previous-version/
  acceptance.md
```

Это учебная структура. brief-current содержит цель и аудиторию; changes - изменения; content-approved - факты и тексты; assets - разрешённые изображения. Предыдущая версия нужна для сравнения, а acceptance описывает проверяемый результат.

Заполненный промпт передачи

> Пересобери сайт по brief-current.md и changes.md. previous-version используй как исходный результат, не как источник актуальных решений. Сохрани подтверждённые факты и фирменные материалы. Сначала сопоставь каждый новый блок с целью из брифа, затем реализуй локальную версию. Не выдумывай отзывы и статистику. После сборки проверь ссылки, телефонный экран и основной сценарий.

Не предполагаю, что один инструмент автоматически видит историю другого. Передача должна состоять из доступных файлов или разрешённых рабочих копий. Если есть только экспортированный сайт, это тоже нужно явно назвать.

Что меняется в роли автора

Я принимаю решения о цели и содержании, а инструмент выполняет пересборку и показывает проверки. Если в исходнике есть устаревшее обещание, технически корректный перенос всё равно будет содержательной ошибкой.

> **Проверка передачи: новый исполнитель может назвать актуальную цель, отменённое решение и критерий завершения без дополнительного пересказа.**

ВЫПУСК 8 · ШАГ 3

Форма: контракт важнее анимации

Учебный пример для сайта с формой. Он не утверждает, что именно такая форма или API есть в моём кейсе.

> Учебный контракт POST /api/demo-requests
> Вход: request_id, name, email, message.
> 201: заявка сохранена, возвращён request_id.
> 200: повтор того же запроса, возвращён прежний ID.
> 400: неверные поля. 409: тот же ID с другим содержимым.
> 5xx или таймаут: результат нужно уточнить, не обещать доставку.

```
const r = await fetch('/api/demo-requests', {
  method: 'POST',
  headers: {'Content-Type': 'application/json'},
  body: JSON.stringify(payload)
});
if (!r.ok) throw new Error('REQUEST_FAILED');
const receipt = await r.json();
```

Это фрагмент внутри асинхронного обработчика, не готовый сайт. Один request_id сохраняется на всю логическую заявку: при повторном клике и таймауте он тот же. Сервер атомарно обеспечивает уникальность ID и отклоняет другое содержимое с прежним ID. Ошибка сервера сама по себе не отклоняет Promise fetch: статус ответа проверяется явно.

После ответа проверяю наличие корректного request_id и показываю подтверждение сохранения. Если уведомление ответственному отправляется отдельно, его статус нельзя автоматически называть успешным. Ключи доступа к почте или боту остаются на сервере.

Промпт: проверить обработчик

> Реализуй учебный контракт выше на тестовом приёмнике. Не подключай рабочие рассылки. Проверь успех, неверный email, повтор ID и таймаут после сохранения. Верни код, способ запуска и журнал ожидаемых исходов. Не показывай пользователю «доставлено», если подтверждено только сохранение.

MDN: fetch и проверка HTTP-статуса

ВЫПУСК 8 · ШАГ 4

Приёмка начинается с плохого сценария

Набор проверок для учебной формы; в реальном сайте подставляется его основной сценарий.

F1: корректный ввод → сохранённая заявка и её ID.
F2: пустое обязательное поле → объяснение рядом с полем.
F3: двойной клик → одна заявка.
F4: сервер сохранил, ответ потерян → сверка по ID, без слепого дубля.
F5: сервер недоступен → честное сообщение; текст формы сохранён.
F6: клавиатура и узкий экран → весь путь доступен.

Проверка браузера удобна для человека, серверная проверка нужна независимо от неё. Клиент можно обойти. Для учебного примера важно хотя бы убедиться, что сервер не принимает пустой объект вместо заявки.

Что проверяется в двух местах

В браузере - сообщение и состояние формы. В приёмнике - фактическая запись, уникальный ID и отсутствие дубля. Только первый экран не доказывает доставку; только запись в базе не доказывает понятный пользовательский путь.

Промпт: независимая приёмка

Пройди F1–F6 на тестовой странице и приёмнике. Для каждого результата сохрани вход, наблюдение в браузере и подтверждение на сервере. Отдели выполненные проверки от недоступных. Не отправляй реальные сообщения клиентам. Если невозможно увидеть приёмник, пометь доставку как непроверенную.

При отсутствии доступа я готовлю точную инструкцию владельцу: куда открыть, что ввести, какой ID найти. Просьба «посмотрите, всё ли хорошо» не даёт проверяемого результата.

В отчёте могут одновременно стоять «макет принят» и «доставка не проверена». Это нормальное состояние работы, если оно явно обозначено.

ВЫПУСК 8 · ШАГ 5

Повторить кейс на своём проекте

Быстрый результат опирается на подготовленные решения и заканчивается проверкой новой версии.

Шесть шагов

1. Сохранить предыдущую версию отдельно. 2. Собрать актуальный бриф и изменения. 3. Подготовить тексты и материалы. 4. Передать ограниченную задачу исполнителю. 5. Проверить содержание, адаптивность, ссылки и ключевое действие. 6. Записать новую версию и оставшиеся ограничения.

> Карточка пересборки
> Исходная версия: …
> Что изменилось в проекте: …
> Основание нового решения: …
> Что сохранить: …
> Что убрать или заменить: …
> Как проверить результат: …
> Время подготовки / сборки / правок: …

Промпт: сравнить до и после

> Сопоставь прежний сайт, changes.md и новую версию. Для каждого изменения покажи реализацию и проверку. Найди старые формулировки, которые противоречат новому контексту. Отдели визуальную готовность от работающих интеграций. Верни итоговый пакет, ограничения и порядок публикации; самовольно не меняй домен и доступы.

Форма на предыдущих страницах - самостоятельный технический пример приёмки. Если на вашем сайте её нет, замените его проверкой реального действия: открыть материал, перейти к контакту, посмотреть демонстрацию. Не добавляйте форму ради упражнения.

> **Мой кейс: первая версия в Kimi, затем пересборка в Codex за 30 минут из обновлённого контекста. Переносимый навык - сохранить решения так, чтобы новый инструмент мог быстро продолжить работу.**

В следующем выпуске проверяю ещё один тип результата: исследование и расчёт, где правдоподобное число может скрывать несопоставимые входы.

ПРОДОЛЖИМ РАЗГОВОР

ИТ-Коктель с Анной Фединой

Про ИТ, управление, людей и жизнь вокруг работы.
Выберите площадку, на которой вам удобно.

Telegram
Заметки, обсуждения и новые материалы
Открыть >

ВК Видео
Выпуски и короткие фрагменты
Открыть >

Дзен
Тексты и видео
Открыть >

RuTube
Разговоры и выпуски подкаста
Открыть >

Сетка
Профессиональные темы и общение
Открыть >

Яндекс Музыка
ИТ-коктейль в аудиоформате
Открыть >

Названия площадок и карточки кликабельны.

