Механизм анонсов новых фич

в данной книге описано, как реализован механихм отображения новых возможностей, которые были добавлены в проект разработчиками

- Как это устроено. Для разработчиков

Как это устроено. Для разработчиков

Механизм анонсов новых фич

Назначение

Анонсы новых фич — отдельная подсистема рядом с турами, но не часть механизма туров.

Тур учит пользователя работать со страницей и запускается по кнопке `?`.

Анонс новой фичи — разовое уведомление формата «здесь появилось новое». Он показывается автоматически на нужной странице, привязывается к конкретному элементу интерфейса и после просмотра больше не появляется.

Общее с турами у анонсов только:

- `data-tour` как способ поставить якорь на DOM-элемент;
- `waitForElement()` как механизм ожидания появления якоря;
- Driver.js как библиотека показа.

Всё остальное у анонсов своё:

- отдельный реестр;
- отдельное localStorage-хранилище;
- отдельный Pinia-store прогресса;
- отдельная точка автозапуска;
- отдельный срок жизни.

Как использовать скилл `feature-announcement` для добавления анонса новой фичи.

1. Если этот skill еще не у тебя. Сделай `git pull` в этом репозитории — скилл лежит в `.claude/skills/feature-announcement/SKILL.md` и подтянется автоматически, вручную ничего настраивать не нужно.
2. Открой новую сессию Claude Code в проекте (или просто попроси в чате): *«добавь анонс новой фичи для кнопки X»*, *«нужно анонсировать [фича]»*, *«покажи хинт про новую фичу»* — Claude сам распознает и вызовет скилл.
3. Ответь на три вопроса по ходу диалога:
   - **на какой элемент** повесить анонс (страница (компонент) + конкретный UI-элемент);
   - **card или hint** — если не уверен, Claude подскажет критерий;
   - **текст** — заголовок и описание, при желании отдельная подпись для меню «Что нового».
4. Claude предложит `id` анонса — подтверди его (после продакшена он неизменяем), после чего Claude сам создаст файл, зарегистрирует его и прогонит тесты.
5. Пример ключа в local storage: Ключ: sociometer:feat:marked-poll Значение: {"expiresAt":1789114576162,"seen":true} Важно, если значение "seen" = true, то это означает, что, анонс был уже просмотрен и больше не будет виден. Для тестов нужно удалять этот параметр вручную или в профиле пользователя есть кнопка. позволяющая просмотреть все анонсы заново "Сбросить обучение". Можно пройти в профиль и нажать на эту кнопку, для того, чтобы руками не изменять ничего в local storage. Но при этом будут показаны все туры заново. (в профиле «Сбросить обучение» → зайти на страницу фичи, убедиться, что анонс показался)

Явно называть команду/слэш не обязательно — достаточно попросить обычными словами, скилл триггерится по описанию.

Удаление анонса через скилл `feature-announcement`

Скилл умеет не только заводить анонсы, но и полностью убирать уже существующие (пользователь решил, что больше не нужно его отображать или удалил фичу, и т. п.).

Как вызвать: попросить обычными словами — *«удали анонс фичи X»*, *«убери анонс с заголовком "Список избранных"»*, *«удали `feat-marked-poll`»*.

Что нужно указать: ID анонса (`feat-<slug>`) или его заголовок (`popover.title`). Если из запроса непонятно, какой анонс имеется в виду (текст не нашёлся, нашлось несколько совпадений, или не назвали ни то ни другое) — скилл сам спросит: **«Укажите ID анонса (`feat-...`) или напишите заголовок»**.

Что произойдёт:

1. Скилл найдёт анонс и покажет его на подтверждение (id, заголовок, файл, route) — удаление необратимо в рабочем дереве.
2. После подтверждения удалит файл `src/tour/features/<slug>.js` и вычистит регистрацию (импорт + запись) из `index.js`.
3. Прогонит `npx vitest run src/tour/features/`, чтобы убедиться, что реестр не сломан.

Что НЕ будет тронуто:

- `data-tour="<slug>"` в шаблоне компонента — якорь общий с турами, тот же элемент может использовать тур. Удаляется только по отдельной явной просьбе и после проверки, что нигде больше не используется.
- `localStorage` пользователей, уже видевших анонс (`sociometer:feat:<slug>`) — как и у `npm run features:prune`

Ключевые правила

1. Анонс показывается **автоматически** при попадании пользователя на маршрут, указанный в определении анонса.
2. Анонс живёт для конкретного пользователя **14 дней с момента первой встречи**, а не с даты релиза.
3. Просмотренный анонс больше не показывается.
4. Если анонс не удалось показать — например, не найден якорь или сейчас идёт тур — он **не считается просмотренным** и попробует показаться при следующем заходе.
5. Анонсы не зависят от флага «Больше не показывать туры». Пользователь может отключить обучение, но это не отключает новости продукта.
6. Просроченные анонсы из кода удаляются отдельным npm-скриптом.

Задействованные файлы

Основные файлы механизма

Файл	Назначение
`/src/tour/features/README.md`	Документация для разработчика: чек-лист, формат анонса, правила регистрации и очистки
`/src/tour/features/index.js`	Реестр всех активных анонсов, функции отбора по маршруту и для будущего меню «Что нового»
`/src/tour/featureStorage.js`	Работа с `localStorage`: ключи `sociometer:feat:*`, чтение, запись, удаление, TTL 14 дней
`/src/tour/featureProgress.js`	Pinia-store прогресса анонсов: создание записи, проверка просмотра/истечения, отметка `seen`, сброс
`/src/tour/runPendingFeatureAnnouncements.js`	Основная политика автопоказа: отбор анонсов, ожидание якорей, запуск карточек/хинтов, отметка просмотра
`/src/tour/featureHints.js`	Показ анонсов типа `hint` через `driver.js/hints`
`/src/tour/featureHints.css`	Стили маячка `Новое` для `hint`-анонсов
`/src/tour/featurePrune.js`	Чистая логика определения просроченных анонсов и удаления регистрации из реестра
`/scripts/pruneFeatureAnnouncements.js`	CLI-скрипт `npm run features:prune` для удаления просроченных анонсов из кода
`/src/App.vue`	Единая точка запуска анонсов на смену маршрута и при появлении авторизации
`/src/pages/profile.vue`	Кнопка «Сбросить обучение», которая сбрасывает и прогресс туров, и прогресс анонсов

Тесты

Файл	Что проверяет
`/src/tour/featureStorage.test.js`	Префиксы ключей, TTL, чтение/запись/удаление записей анонсов
`/src/tour/featureProgress.test.js`	Поведение Pinia-store: `ensureRecord`, `markSeen`, `isExpired`, `resetAll`, `revision`
`/src/tour/features/registry.test.js`	Отбор анонсов по маршруту, сортировку по `addedAt`, данные для будущего меню
`/src/tour/features/featureIds.test.js`	Валидацию реальных анонсов: id, route, addedAt, kind, popover, label, регистрацию в реестре
`/src/tour/runPendingFeatureAnnouncements.test.js`	Политику показа карточек и хинтов
`/src/App.featureAnnouncements.test.js`	Интеграцию в `App.vue`: watch маршрута, запуск после авторизации, retry, снятие хинтов

Пример текущего анонса

Файл:

```text
/src/tour/features/marked-poll.js
```

Содержимое:

```js
export default {
  id: "feat-marked-poll",
  addedAt: "2026-08-28",
  route: "POLLS",
  kind: "hint",
  element: '[data-tour="marked-poll"]',
  popover: {
    title: "Список избранных",
    description: "Теперь можно добавлять опросы в список избранных",
  },
  label: "Избранные опросы",
};
```

Якорь для него стоит в:

```text
/src/pages/pollList.vue
```

```vue
<div
  class="polls_filter__btn polls_list__hdr-btn"
  data-tour="marked-poll"
  :class="{ 'polls_list__hdr-btn--on': favoriteOnly }"
  @click="toggleFavoriteOnly()"
>
```

Как работает механизм

1. Разработчик ставит якорь на элемент интерфейса

Анонс должен быть привязан к конкретному DOM-элементу. Для этого на элемент ставится атрибут:

```vue
data-tour="<slug>"
```

Например:

```vue
<div data-tour="marked-poll">
  ...
</div>
```

Важно: якорь нужно ставить именно на тот элемент, около которого должен появиться анонс, а не на случайную обёртку и не на внутренний элемент библиотеки Element Plus.

Обычно селектор в анонсе выглядит так:

```js
element: '[data-tour="marked-poll"]'
```

---

2. Анонс описывается отдельным файлом

Каждый анонс — это отдельный файл в папке:

```text
/src/tour/features/
```

Имя файла — slug анонса:

```text
/src/tour/features/<slug>.js
```

Например:

```text
/src/tour/features/marked-poll.js
```

Файл должен экспортировать **только данные**, без логики:

```js
export default {
  id: "feat-marked-poll",
  addedAt: "2026-08-28",
  route: "POLLS",
  kind: "hint",
  element: '[data-tour="marked-poll"]',
  popover: {
    title: "Список избранных",
    description: "Теперь можно добавлять опросы в список избранных",
  },
  label: "Избранные опросы",
};
```

---

3. Анонс регистрируется в общем реестре

Реестр находится в файле:

```text
/src/tour/features/index.js
```

Пример:

```js
import { useFeatureProgress } from "@/tour/featureProgress.js";
import markedPoll from "./marked-poll.js";

export const featureAnnouncements = {
  markedPoll,
};
```

Важно соблюдать формат регистрации:

```js
import someFeature from "./some-feature.js";

export const featureAnnouncements = {
  someFeature,
};
```

Почему это важно:

- скрипт очистки `npm run features:prune` удаляет просроченные анонсы из реестра построчно;
- он ожидает понятный формат импорта и регистрации;
- если регистрация будет записана слишком сложно или растянута на несколько строк, скрипт откажется работать и ничего не удалит.

---

4. `App.vue` автоматически запускает проверку анонсов

Разработчику **не нужно** вручную вызывать запуск анонса в `created()` страницы.

Единая точка запуска находится в:

```text
/src/App.vue
```

Там импортируется:

```js
import {
  runPendingFeatureAnnouncements,
  FEATURE_SKIP_TOUR_ACTIVE,
} from "@/tour/runPendingFeatureAnnouncements.js";
```

`App.vue` следит за изменением имени маршрута:

```js
watch: {
  "$route.name": {
    immediate: true,
    handler(name, oldName) {
      ...
      this.runFeatureAnnouncements(name);
    },
  },
}
```

Также есть отдельный watcher на авторизацию:

```js
"useStoreProfile.isAuth"(isAuth) {
```

```
  if (isAuth) this.runFeatureAnnouncements(this.$route?.name);
}
```

Это нужно для ситуации, когда пользователь попал в приложение сразу по ссылке, но на момент первого route-watch ещё не был авторизован.

---

5. `runPendingFeatureAnnouncements()` выбирает подходящие анонсы

Основная логика находится в:

```text
/src/tour/runPendingFeatureAnnouncements.js
```

Функция получает имя маршрута:

```js
runPendingFeatureAnnouncements(routeName, { router, now, registry })
```

Дальше она:

1. берёт анонсы для текущего маршрута через `announcementsForRoute()`;
2. отбрасывает уже просмотренные;
3. отбрасывает истёкшие для пользователя;
4. создаёт запись в localStorage, если пользователь встретил анонс впервые;
5. ждёт появления DOM-якоря через `waitForElement()`;
6. показывает анонс как `card` или `hint`;
7. отмечает анонс просмотренным в нужный момент.

Отбор анонсов по маршруту

Файл:

```text
/src/tour/features/index.js
```

Функция:

```js
export function announcementsForRoute(routeName, registry = featureAnnouncements) {
  return Object.values(registry)
    .filter((announcement) => matchesRoute(announcement, routeName))
    .sort((a, b) => String(a.addedAt).localeCompare(String(b.addedAt)));
}
```

`route` в анонсе может быть строкой:

```js
route: "POLLS"
```

или массивом строк:

```js
route: ["POLLS", "RESULTS"]
```

Если на одном маршруте несколько анонсов, они сортируются по `addedAt`: более старые показываются первыми.

Хранилище прогресса

Ключи localStorage

Файл:

```text
/src/tour/featureStorage.js
```

Константы:

```js
export const FEATURE_ID_PREFIX = "feat-";
export const FEATURE_KEY_PREFIX = "sociometer:feat:";
export const FEATURE_TTL_MS = 14 * 24 * 60 * 60 * 1000;
```

Для анонса:

```js
id: "feat-marked-poll"
```

ключ в localStorage будет:

```text
sociometer:feat:marked-poll
```

Значение:

```json
{
  "expiresAt": 1800000000000,
  "seen": false
}
```

Почему `id` начинается с `feat-`

Префикс `feat-` — единственный признак, по которому приложение отличает анонс новой фичи от шагов обычных туров.

Правильно:

```js
id: "feat-results-pdf-export"
```

Неправильно:

```js
id: "results-pdf-export"
```

После выхода анонса в прод `id` нельзя переименовывать: это часть ключа в браузере пользователя. Переименование будет воспринято как новый анонс, и пользователи увидят его повторно.

Срок жизни анонса для пользователя

Анонс показывается конкретному пользователю максимум **14 дней с момента первой встречи**.

Это делает `featureProgress.ensureRecord()`:

```js
ensureRecord(id, now = Date.now()) {
  const existing = getFeatureRecord(id);
  if (existing) return existing;
  const record = { expiresAt: now + FEATURE_TTL_MS, seen: false };
  setFeatureRecord(id, record);
  this.revision++;
  return record;
}
```

Важный момент: срок отсчитывается не от `addedAt`, а от момента, когда пользователь впервые оказался на маршруте, где мог увидеть анонс.

Например:

- фича вышла 1 августа;
- пользователь впервые зашёл на нужную страницу 10 августа;
- его личное окно показа закончится 24 августа.

Типы анонсов: `card` и `hint`

Поле `kind` определяет способ показа:

```js
kind: "card"
```

или:

```js
kind: "hint"
```

`card`

`card` — это карточка Driver.js поверх страницы. Используется, когда фичу нужно объяснить текстом.

Пример:

```js
export default {
  id: "feat-results-pdf-export",
  addedAt: "2026-08-28",
  route: "RESULTS",
  kind: "card",
  element: '[data-tour="results-pdf-export"]',
  popover: {
    title: "Экспорт в PDF",
    description: "Теперь результаты опроса можно выгрузить в PDF.",
  },
  label: "Экспорт результатов в PDF",
};
```

Показывается через `runTour()` из `tourEngine.js`, но как одношаговый тур:

```js
await runTour(
  {
    id: announcement.id,
    steps: [
      {
        id: announcement.id,
        element: announcement.element,
        popover: announcement.popover,
        optional: announcement.optional,
      },
    ],
  },
  {
    router,
    onShown: () => {
      progress.markSeen(announcement.id);
      shown.push(announcement.id);
    },
  }
);
```

Анонс типа `card` считается просмотренным **в момент появления карточки на экране**, а не при закрытии.

Почему так: если пользователь уйдёт со страницы или произойдёт программный редирект, promise `runTour()` может не завершиться. Если ждать закрытия, анонс мог бы показываться снова и снова.

`hint`

`hint` — это маячок «Новое» около элемента. Используется, когда фича понятна сама по себе и не нужно перекрывать экран карточкой.

Пример:

```js
export default {
  id: "feat-marked-poll",
  addedAt: "2026-08-28",
  route: "POLLS",
  kind: "hint",
  element: '[data-tour="marked-poll"]',
  popover: {
    title: "Список избранных",
    description: "Теперь можно добавлять опросы в список избранных",
  },
  label: "Избранные опросы",
};
```

Показывается через:

```text
/src/tour/featureHints.js
```

и `driver.js/hints`.

Визуально это бейдж:

```text
Новое
```

Стили находятся в:

```text
/src/tour/featureHints.css
```

Анонс типа `hint` считается просмотренным **только после закрытия конкретного хотспота** — например, после нажатия «Понятно».

Причина: сам факт появления маячка ещё не гарантирует, что пользователь его увидел. Его могла перекрыть модалка, открывшаяся следующим тиком.

Ожидание якоря

Анонс ждёт DOM-элемент через:

```text
/src/tour/waitForElement.js
```

В `runPendingFeatureAnnouncements.js` используются таймауты:

```js
export const ANCHOR_TIMEOUT_MS = 8000;
export const OPTIONAL_ANCHOR_TIMEOUT_MS = 100;
```

Обычный анонс ждёт якорь до 8 секунд.

Если у анонса есть:

```js
optional: true
```

якорь ждётся только 100 мс.

`optional: true` нужен для ситуаций, когда элемент может структурно отсутствовать:

- пустой список;
- отключённый режим;
- элемент доступен только при определённом состоянии страницы.

Если якорь не найден, анонс не помечается просмотренным.

Защита от конфликтов с турами и хинтами

Если уже идёт тур

Для `hint`-анонсов есть проверка:

```js
if (pendingHints.length > 0 && isTourActive()) {
  return markSkippedBecause(shown, FEATURE_SKIP_TOUR_ACTIVE);
}
```

Если сейчас активен тур, хинт-анонс не показывается и не помечается просмотренным.

В `App.vue` для такого случая есть retry: если анонс был пропущен из-за активного тура, запуск повторяется позже.

Если открыт мобильный drawer

`featureHints.js` содержит:

```js
pauseFeatureHintsForDrawer()
resumeFeatureHintsForDrawer()
```

Это нужно, чтобы маячки анонсов не просвечивали поверх мобильного бокового меню.

Если пользователь нажал кнопку `?`

В `App.vue` перед показом обычных подсказок/туров вызывается:

```js
dismissFeatureHintsBeforeHelp()
```

Если на экране есть анонс-хинт, он сначала снимается. Иначе два разных набора хинтов могли бы конфликтовать.

При смене маршрута

`App.vue` гасит показанный анонс-хинт при уходе со страницы, потому что у анонса нет компонента-владельца страницы, который мог бы сделать teardown в `beforeUnmount`.

Формат определения анонса

Обязательные поля:

Поле	Тип	Назначение
`id`	`string`	Идентификатор в формате `feat-<slug>`
`addedAt`	`string`	Дата релиза в формате `YYYY-MM-DD`; используется для очистки из кода
`route`	`string | string[]`	Имя маршрута из `src/routes.js`, где показывать анонс
`kind`	`"card" | "hint"`	Тип показа
`element`	`string`	CSS-селектор якоря
`popover.title`	`string`	Заголовок карточки/хинта
`popover.description`	`string`	Описание; пустым быть не должно
`label`	`string`	Короткое название для будущего меню «Что нового»

Необязательные поля:

Поле	Тип	Назначение
`pruneAfterDays`	`number`	Сколько дней анонс живёт в коде; по умолчанию 28
`optional`	`boolean`	Если `true`, якорь может отсутствовать, и его ждут коротким таймаутом
`beacon`	`object`	Дополнительные настройки beacon для `hint`
`popover.popoverClass`	`string`	Дополнительный CSS-класс popover

Что нужно сделать разработчику, чтобы опубликовать анонс

Шаг 1. Выбрать slug

Slug должен быть стабильным и понятным.

Примеры:

```text
marked-poll
results-pdf-export
dashboard-new-chart
```

Из slug формируются:

- имя файла;
- `data-tour`;
- `id`;
- localStorage-ключ.

Например для slug:

```text
results-pdf-export
```

получается:

```text
Файл: src/tour/features/results-pdf-export.js
id: feat-results-pdf-export
data-tour: results-pdf-export
localStorage: sociometer:feat:results-pdf-export
```

Шаг 2. Поставить якорь на элемент

В нужном Vue-компоненте добавить:

```vue
data-tour="results-pdf-export"
```

Пример:

```vue
<el-button data-tour="results-pdf-export">
  Экспорт в PDF
</el-button>
```

Важно:

- не ставить якорь на скрытый элемент;
- не ставить якорь на нестабильную внутреннюю разметку Element Plus;
- убедиться, что элемент реально есть на маршруте, указанном в `route`.

Шаг 3. Создать файл анонса

Создать файл:

```text
/src/tour/features/results-pdf-export.js
```

Пример для карточки:

```js
export default {
  id: "feat-results-pdf-export",
  addedAt: "2026-08-28",
  route: "RESULTS",
  kind: "card",
  element: '[data-tour="results-pdf-export"]',
  popover: {
    title: "Экспорт в PDF",
    description: "Теперь результаты опроса можно выгрузить в PDF-файл.",
  },
  label: "Экспорт результатов в PDF",
};
```

Пример для хинта:

```js
export default {
  id: "feat-results-pdf-export",
  addedAt: "2026-08-28",
  route: "RESULTS",
  kind: "hint",
  element: '[data-tour="results-pdf-export"]',
  popover: {
    title: "Экспорт в PDF",
    description: "Теперь результаты опроса можно выгрузить в PDF-файл.",
  },
  label: "Экспорт результатов в PDF",
};
```

Шаг 4. Зарегистрировать анонс

Открыть:

```text
/src/tour/features/index.js
```

Добавить импорт:

```js
import resultsPdfExport from "./results-pdf-export.js";
```

Добавить запись в `featureAnnouncements`:

```js
export const featureAnnouncements = {
  markedPoll,
  resultsPdfExport,
};
```

Рекомендуемый формат:

```js
import resultsPdfExport from "./results-pdf-export.js";

export const featureAnnouncements = {
  resultsPdfExport,
};
```

Не растягивать одну регистрацию на несколько строк.

Плохо:

```js
export const featureAnnouncements = {
  "feat-results-pdf-export":
    // комментарий
    resultsPdfExport,
};
```

Шаг 5. Проверить тестами

Запустить все тесты:

```bash
npm test
```

Или только тесты анонсов:

```bash
npx vitest run src/tour/features/
```

Минимально полезная проверка механизма:

```bash
npx vitest run src/tour/featureStorage.test.js src/tour/featureProgress.test.js
src/tour/features/registry.test.js src/tour/features/featureIds.test.js
src/tour/runPendingFeatureAnnouncements.test.js src/App.featureAnnouncements.test.js
```

`featureIds.test.js` поймает:

- забытый префикс `feat-`;
- дублирующийся `id`;
- несуществующий `route`;
- неправильный `addedAt`;
- неизвестный `kind`;
- пустой `element`;
- пустой `label`;
- пустой `popover.title`;
- пустой `popover.description`;
- файл анонса, который забыли зарегистрировать в `index.js`.

Шаг 6. Проверить руками

1. Перейти в профиль.
2. Нажать кнопку **«Сбросить обучение»**.
3. Открыть страницу, указанную в `route`.
4. Убедиться, что анонс появился.
5. Обновить страницу или зайти снова.
6. Убедиться, что анонс больше не появляется.

Кнопка «Сбросить обучение» находится в:

```text
/src/pages/profile.vue
```

Она вызывает:

```js
useTourProgress().resetAll(Object.keys(tours));
useFeatureProgress().resetAll();
```

То есть сбрасывает и туры, и анонсы новых фич.

Очистка просроченных анонсов из кода

Анонс живёт в коде по умолчанию **28 дней** с даты `addedAt`.

Это задаётся в:

```text
/src/tour/featurePrune.js
```

```js
export const DEFAULT_PRUNE_AFTER_DAYS = 28;
```

Почему 28 дней:

- 14 дней — чтобы пользователь вообще успел зайти в систему после релиза;
- ещё 14 дней — его личное окно показа с момента первой встречи.

Если нужно изменить срок жизни конкретного анонса в коде:

```js
export default {
  id: "feat-some-feature",
  addedAt: "2026-08-28",
  pruneAfterDays: 45,
  ...
};
```

Проверить просроченные анонсы

```bash
npm run features:prune
```

```

Без `--apply` это dry-run:

- печатает список просроченных анонсов;
- ничего не удаляет;
- если есть просроченные анонсы, выходит с ненулевым кодом.

---

### Удалить просроченные анонсы

```bash
npm run features:prune -- --apply
```

Скрипт:

1. находит просроченные файлы в `/src/tour/features/`;
2. проверяет, что может безопасно удалить регистрацию из `index.js`;
3. удаляет файл анонса;
4. удаляет импорт и запись из `featureAnnouncements`.

Если формат регистрации непонятный, скрипт откажется работать и ничего не удалит.

Что делать не нужно

Не нужно добавлять вызов анонса в `created()` страницы.

Не нужно трогать:

```text
/src/tour/tours/
```

Не нужно добавлять анонс в `ROUTE_TOURS`.

Не нужно вызывать:

```js
autoRunPendingSteps()
```

Не нужно придумывать новый ключ localStorage.
```

Не нужно переименовывать `id` анонса после выхода в прод.

Не нужно удалять просроченные анонсы руками — для этого есть:

```bash
npm run features:prune -- --apply
```

Если анонс не показался

Проверять по порядку:

1. Пользователь авторизован? На экране логина анонсы не показываются.
2. Совпадает ли `route` с текущим `$route.name`?
3. Есть ли элемент с нужным `data-tour` в DOM?
4. Не скрыт ли элемент?
5. Не стоит ли в localStorage запись `sociometer:feat:<slug>` с `seen: true`?
6. Не истёк ли `expiresAt`?
7. Не идёт ли сейчас обычный тур?
8. Не забыли ли зарегистрировать файл в `/src/tour/features/index.js`?
9. Проходит ли `npx vitest run src/tour/features/`?

Для ручной перепроверки проще всего нажать **«Сбросить обучение»** в профиле.

Краткий чек-лист публикации

1. Выбрать slug.
2. Поставить `data-tour="<slug>"` на элемент фичи.
3. Создать файл `/src/tour/features/<slug>.js`.
4. Указать `id: "feat-<slug>"`.
5. Указать корректный `addedAt`.
6. Указать существующий `route` из `src/routes.js`.
7. Выбрать `kind: "card"` или `kind: "hint"`.
8. Заполнить `element`, `popover.title`, `popover.description`, `label`.
9. Зарегистрировать анонс в `/src/tour/features/index.js`.
10. Запустить тесты.
11. Сбросить обучение в профиле и проверить показ руками.
12. После истечения срока жизни в коде удалить через `npm run features:prune -- --apply`.
