Ай-ТИ Фреш

ТЕХНИЧЕСКИЙ РАЗБОР

Оптимизация фронтенда: как мы снижаем время загрузки

Наша методология Core Web Vitals: code splitting, AVIF, critical CSS, Workbox, INP-контроль

Июль 2026

itfresh.ru · ИТ-аутсорсинг для юридических лиц

О ЧЁМ ЭТОТ РАЗБОР

Суть проблемы

Медленный фронтенд бьёт по деньгам: при LCP выше 4 с и INP выше 500 мс страница попадает в «плохую» зону Core Web Vitals, теряет позиции в поиске и конверсию. Мы разбираем, как системно снижаем время загрузки — от анализа бандла и ленивой загрузки чанков до AVIF-пайплайна, critical CSS и Service Worker — и закрепляем результат полевым мониторингом на 75-м перцентиле реальных пользователей.

Почему это важно бизнесу

- Google оценивает Core Web Vitals по полю (75-й перцентиль реальных визитов) — медленный сайт проседает в ранжировании и трафике.
- Каждая лишняя секунда LCP и каждые 100 мс INP на мобильных — это отказы в корзине и упущенные заявки.
- Тяжёлый JS-бандл на 3G/4G съедает батарею и терпение: рост bounce rate прямо коррелирует с весом первой загрузки.
- Разовая оптимизация деградирует за релизы — без бюджета на бандл в CI регресс возвращается через пару спринтов.

ЧТО МЫ ЗАКЛАДЫВАЕМ

Ключевые параметры реализации

≤2.5 с

Порог LCP «good» по Core Web Vitals — целимся ниже, с запасом на p75 мобильных

web.dev / CWV

≤200 мс

Порог INP «good»: INP с 12.03.2024 заменил FID, именно его мы мониторим

web.dev / CWV

≤0.1

Порог CLS «good»; держим width/height и font size-adjust, чтобы не прыгала вёрстка

web.dev / CWV

0.35.1

sharp: наш конвейер AVIF/WebP; AVIF quality 50, effort 4, subsampling 4:4:4

по докам sharp

7.4.x

Workbox: precache + runtime-кеш (CacheFirst для картинок, SWR для API)

по докам Workbox

5 КБ

Порог прироста бандла на PR — выше требуем обоснование в CI

наш стандарт

РЕАЛИЗАЦИЯ 1 ИЗ 3

Разбор бандла и code splitting по роутам

Что настраиваем

Клиентский React-19 SPA: главная тянула единый бандл со страницами бронирования, ЛК и админки

Как мы это делаем

1. Снимаем карту бандла webpack-bundle-analyzer (analyzerMode: static), находим тяжёлые узлы: moment с локалями, mapbox-gl, полный lodash на каждой странице
2. Режем по роутам через React.lazy(() => import()) + <Suspense fallback>, каждый маршрут — отдельный чанк, подгружается по требованию
3. Заменяем moment на date-fns v4 (tree-shakeable), lodash — на точечные lodash/debounce, lodash/groupBy вместо корневого импорта
4. Ставим sideEffects: ['*.css','*.scss'] в package.json и убираем barrel-index, из-за которого импорт Button тянул MapView

РЕЗУЛЬТАТ

Бандл главной падает с мегабайтов до сотен килобайт (gzip): первая отрисовка перестаёт ждать код чужих страниц. LCP и INP на входе улучшаются сразу — главный поток не парсит лишний JS.

КЛЮЧЕВОЙ НЮАНС

Barrel-файлы (index.ts с реэкспортом) молча ломают tree shaking: один именованный импорт тянет весь модуль. Смотрим в доку webpack по sideEffects и импортируем из конкретных файлов.

РЕАЛИЗАЦИЯ 2 ИЗ 3

AVIF/WebP-пайплайн и защита от CLS

Что настраиваем

40+ изображений отелей в неоптимизированном PNG без адаптива под размер экрана и DPR

Как мы это делаем

1. Скриптом на sharp 0.35.1 генерируем 3 ширины (400/800/1200) в трёх форматах: AVIF (quality 50, effort 4), WebP (quality 75), JPEG-fallback
2. Отдаём через <picture>: type image/avif → image/webp → <img>, атрибуты srcset+sizes для выбора по вьюпорту
3. На каждый <img> ставим width/height (или aspect-ratio), loading=lazy и decoding=async — CLS от перекомпоновки уходит в ноль
4. withoutEnlargement: true в resize, чтобы sharp не растягивал мелкие оригиналы и не плодил мыло

РЕЗУЛЬТАТ

Средний вес карточки падает в разы (PNG→AVIF), трафик и число запросов резко сокращаются. Резерв размеров убирает скачки макета при догрузке — CLS держится в «good» на медленных сетях.

КЛЮЧЕВОЙ НЮАНС

AVIF-параметр effort (0–9, дефолт 4) — это баланс «размер vs время сборки»: на CI-конвейере поднимаем осторожно, иначе билд картинок растягивается в разы при копеечном выигрыше веса.

РЕАЛИЗАЦИЯ 3 ИЗ 3

Critical CSS, шрифты и Service Worker

Что настраиваем

Блокирующий CSS ~180 КБ, Google Fonts 5 начертаний кириллицы, нулевой кеш повторных визитов

Как мы это делаем

1. Инлайним critical CSS в <head>, остальной грузим через rel=preload as=style onload=this.rel='stylesheet' + <noscript>-fallback
2. Генерацию above-the-fold мигрировали с архивного critters на его преемник beasties (pruneSource, preload swap)
3. Шрифты: self-host woff2, unicode-range только кириллица+латиница, font-display: swap, size-adjust против FOUT-сдвига; с 5 файлов до 2
4. Повторные визиты — Workbox 7.4: precacheAndRoute статики, CacheFirst для image (ExpirationPlugin), StaleWhileRevalidate для /api/catalog

РЕЗУЛЬТАТ

Первый экран рисуется, не дожидаясь всего CSS и шрифтов; повторный заход поднимается из кеша Service Worker, данные каталога обновляются в фоне. Immutable-кеш на хешах снимает лишние ревалидации.

КЛЮЧЕВОЙ НЮАНС

critters заархивирован и ушёл в Nuxt-команду — на новых проектах ставим beasties, конфиг совместим. font-display: swap без size-adjust/ascent-override сам порождает CLS при подмене шрифта.

ЧТО МЫ ВЫУЧИЛИ НА ПРАКТИКЕ

Подводные камни

- ✕ **Ориентир на FID вместо INP**
  FID отменён 12.03.2024, его нет в Search Console. Оптимизировать «отклик первого клика» бессмысленно — целимся в INP ≤ 200 мс по всей сессии.
- ✕ **Вера в лабораторный Lighthouse**
  Оценка Google идёт по полю на 75-м перцентиле реальных визитов (CrUX). Зелёный Lighthouse на ноутбуке не гарантирует «good» на бюджетных мобильных.
- ✕ **Barrel-файлы убивают tree shaking**
  index.ts с реэкспортом заставляет бандлер тянуть весь модуль на один импорт. Лечим прямыми импортами и sideEffects: false для не-CSS.
- ✕ **font-display: swap без size-adjust**
  swap показывает текст сразу, но подмена fallback→web-шрифт двигает вёрстку и раздувает CLS. Выравниваем метрики через size-adjust/ascent-override.
- ✕ **AVIF effort на максимум в CI**
  effort 9 экономит проценты веса, но кратно удлиняет сборку картинок. Держим дефолтные 4 и тюним только на реально тяжёлых ассетах.
- ✕ **critters на новых проектах**
  Пакет заархивирован и обновлений не будет. Ставим преемник beasties — конфигурация та же, но с активной поддержкой и фиксами.
- ✕ **immutable без хеша в имени**
  Cache-Control: immutable на файле без contenthash — это залипший старый ассет у клиента. immutable ставим только на main.<hash>.js.
- ✕ **Нет бюджета бандла в CI**
  Без gate на прирост размера регресс возвращается за пару релизов. Ставим порог (напр. +5 КБ на PR) с обязательным обоснованием.

УРОВНИ РЕАЛИЗАЦИИ

Как правильно

МИНИМУМ

- Сжать картинки в WebP, проставить width/height и loading=lazy на все <img>
- Убрать блокирующую загрузку шрифтов: self-host woff2 + font-display: swap
- Включить gzip/brotli и long-cache на статику с contenthash в имени

НОРМАЛЬНО

- Code splitting по роутам (React.lazy+Suspense), moment/lodash → tree-shakeable
- AVIF+WebP через sharp с srcset/sizes, critical CSS через beasties
- Настроить web-vitals RUM и следить за LCP/INP/CLS на p75

ХОРОШО

- Service Worker (Workbox): precache + CacheFirst/SWR, prefetch чанков на hover
- CDN с точками в РФ, immutable-кеш, preconnect/preload критичных ресурсов
- Бюджет бандла в CI + алерты при выходе поля за пороги Core Web Vitals

ПРОВЕРЬТЕ У СЕБЯ

Чек-лист самопроверки

- [ ] Есть ли актуальная карта бандла (webpack-bundle-analyzer) и известны ли самые тяжёлые зависимости?
- [ ] Разбит ли код на чанки по роутам, не тянет ли главная код чужих страниц?
- [ ] Настроен ли sideEffects в package.json и убраны ли barrel-файлы, ломающие tree shaking?
- [ ] Все ли <img> имеют width/height, loading=lazy и отдаются в AVIF/WebP через <picture>?
- [ ] Инлайнится ли critical CSS, а остальной грузится асинхронно (beasties, не архивный critters)?
- [ ] Self-host ли шрифты с unicode-range, font-display: swap и защитой от CLS через size-adjust?
- [ ] Есть ли Service Worker с раздельными стратегиями кеша для статики и API?
- [ ] Мониторим ли мы INP (а не отменённый FID) и смотрим ли поле на 75-м перцентиле?
- [ ] Стоит ли immutable-кеш только на файлах с contenthash в имени?
- [ ] Есть ли в CI бюджет бандла, который блокирует незамеченный рост размера?

Если хотя бы на два вопроса ответ «нет» или «не знаю» — тема требует внимания.

ЧТО МЫ ДЕЛАЕМ

Как поможет ITFresh

ITFresh — ИТ-аутсорсинг для юридических лиц до 50 рабочих мест в Москве и области. 15+ лет практики, собственная инфраструктура в дата-центре МТС (8 серверов Dell Xeon Platinum).

- Проводим аудит фронтенда: Lighthouse + поле CrUX, разбор бандла, план по LCP/INP/CLS
- Внедряем code splitting, AVIF/WebP-пайплайн на sharp, critical CSS и self-host шрифтов
- Настраиваем Service Worker на Workbox и CDN с immutable-кешем под аудиторию в РФ
- Ставим RUM-мониторинг web-vitals и бюджет бандла в CI, чтобы результат не деградировал

15+
лет в ИТ-поддержке

50
рабочих мест — наш профиль

МТС
дата-центр, Москва

КОНТАКТЫ

Обсудить вашу задачу

Сайт	**itfresh.ru**
Телефон	**+7 903 729-62-41**
Telegram	**@ITfresh_Boss**

Бесплатно посмотрим вашу инфраструктуру по этому чек-листу и скажем, где тонко — без обязательств.

itfresh.ru

ДОКУМЕНТАЦИЯ И НАШ СТЕК

Техническая база

01 Web Vitals: LCP, INP, CLS и пороги (web.dev — 2024+)

02 Introducing INP to Core Web Vitals (developers.google.com — 2024)

03 web-vitals: onLCP/onINP/onCLS, attribution (github.com/GoogleChrome — v5.3)

04 sharp: Output options (avif/webp) (sharp.pixelplumbing.com — 0.35)

05 Workbox: strategies и expiration (developer.chrome.com — v7.4)

06 beasties (преемник critters) (github.com/danielroe — 2024+)

07 React: Code-Splitting, lazy и Suspense (react.dev — v19)

08 webpack: sideEffects и tree shaking (webpack.js.org — 5.x)

Основано на официальной документации продуктов и нашей практике внедрения.