Программирование в 2026: как использовать кастомные метрики производительности для поиска узких мест

24.11.2026 · PDF-версия статьи с сайта cms.dolinais.ru

В 2026 году производительность веб-приложений остаётся критическим фактором успеха. Однако стандартные метрики **Web Vitals** (LCP, INP, CLS) не всегда отражают реальные проблемы пользователей. В этой статье разберём, как использовать кастомные метрики производительности на основе данных реальных пользователей (RUM), чтобы находить узкие места и доказывать эффект оптимизаций.

Что такое кастомные метрики производительности и зачем они нужны

Кастомные метрики производительности — это метрики, которые разработчик определяет самостоятельно для измерения конкретных сценариев взаимодействия пользователя с приложением. Они дополняют стандартные Web Vitals, которые фокусируются на общих аспектах загрузки и отзывчивости.

Стандартные метрики хороши для базового мониторинга, но они не учитывают специфику вашего продукта. Например, для социальной сети важно время до появления первого поста в ленте, для интернет-магазина — время до отображения цены и кнопки «Купить», для SaaS-платформы — время отклика на действие пользователя в сложном редакторе. Кастомные метрики позволяют измерять именно эти критические пути.

Кроме того, Web Vitals измеряются в лабораторных условиях или на ограниченной выборке, тогда как RUM (Real User Monitoring) собирает данные от всех пользователей, включая те, кто работает на старых устройствах или медленном интернете. Это даёт полную картину.

Как выбрать точки измерения

Прежде чем внедрять кастомные метрики, определите ключевые пользовательские сценарии. Задайте себе вопросы:

- Какие действия пользователи совершают чаще всего?
- Где возникают задержки, влияющие на конверсию или удержание?
- Какие шаги можно измерить с помощью Performance API?

Например, для ленты ВКонтакте важно время от клика по ссылке до отображения первого поста. Для этого можно использовать метку performance.mark('feed-start') в момент начала загрузки и performance.mark('feed-rendered') после отрисовки первого элемента. Затем измерьте разницу с помощью performance.measure().

Другой пример: время до интерактивности кнопки «Оформить заказ». Здесь можно отслеживать момент, когда кнопка становится кликабельной, и сравнивать с временем загрузки страницы.

Важно выбирать метрики, которые можно автоматически собирать и агрегировать. Используйте PerformanceObserver для отслеживания событий и отправляйте данные на сервер с помощью navigator.sendBeacon() или Fetch API.

Пример реализации сбора кастомной метрики

Рассмотрим простой пример: измерение времени загрузки списка товаров на странице каталога.

```
// ?????? ????????
performance.mark('catalog-start');

// ????? ????????? ?????? ? ????????? ??????
fetch('/api/products').then(response => response.json()).then(products => {
  renderProducts(products);
  performance.mark('catalog-end');
  performance.measure('catalog-load', 'catalog-start', 'catalog-end');
  const measure = performance.getEntriesByName('catalog-load')[0];
  // ???????? ??????? ?? ??????
  navigator.sendBeacon('/analytics', JSON.stringify({ metric: 'catalog-
load', value: measure.duration }));
});
```

Этот подход позволяет собирать данные о времени загрузки каталога у реальных пользователей. Затем эти данные можно агрегировать и анализировать.

Анализ данных и поиск узких мест

Собранные метрики нужно анализировать в разрезе различных сегментов: тип устройства, скорость соединения, географическое положение, версия браузера. Например, если вы видите, что на мобильных устройствах время загрузки каталога значительно выше, чем на десктопах, стоит оптимизировать изображения или уменьшить размер бандла.

Используйте перцентили (50-й, 75-й, 95-й), чтобы понять распределение. Высокий 95-й перцентиль указывает на проблемы у части пользователей, которые могут быть незаметны в средних значениях.

Сравнивайте метрики до и после оптимизаций, чтобы доказать эффект. Например, после внедрения ленивой загрузки изображений время загрузки каталога снизилось на 20% на мобильных устройствах.

Инструменты для сбора и визуализации

Для сбора RUM-данных можно использовать как собственные решения, так и готовые сервисы. В 2026 году популярны:

- **Google Analytics 4 —** позволяет отправлять кастомные события с метриками.
- **Sentry Performance —** отслеживает производительность и ошибки.
- **Datadog RUM —** мощный инструмент для мониторинга реальных пользователей.
- **Собственный стек** на основе ClickHouse или Prometheus + Grafana.

При выборе инструмента учитывайте стоимость, удобство интеграции и требования к приватности данных.

Заключение

Кастомные метрики производительности — это мощный инструмент для понимания того, как ваше приложение работает у реальных пользователей. Они дополняют Web Vitals и позволяют находить узкие места, которые не видны при стандартном мониторинге. Внедрение RUM и анализ кастомных метрик помогает принимать обоснованные решения об оптимизациях и доказывать их эффективность. Начните с определения ключевых сценариев, выберите метрики, настройте сбор данных и регулярно анализируйте результаты — это окупится улучшением пользовательского опыта и бизнес-показателей.

Источник: https://cms.dolinais.ru/programmirovanie-v-2026-kak-ispolzovat-kastomnye-metriki-proizvoditelnosti-dlya-poiska-uzk