Как мы ускорили мобильную версию ВКонтакте в 4 раза

Обо мне

- Team Lead команды Core [m.vk.com](m.vk.com)
- 6 лет занимаюсь m.vk.com

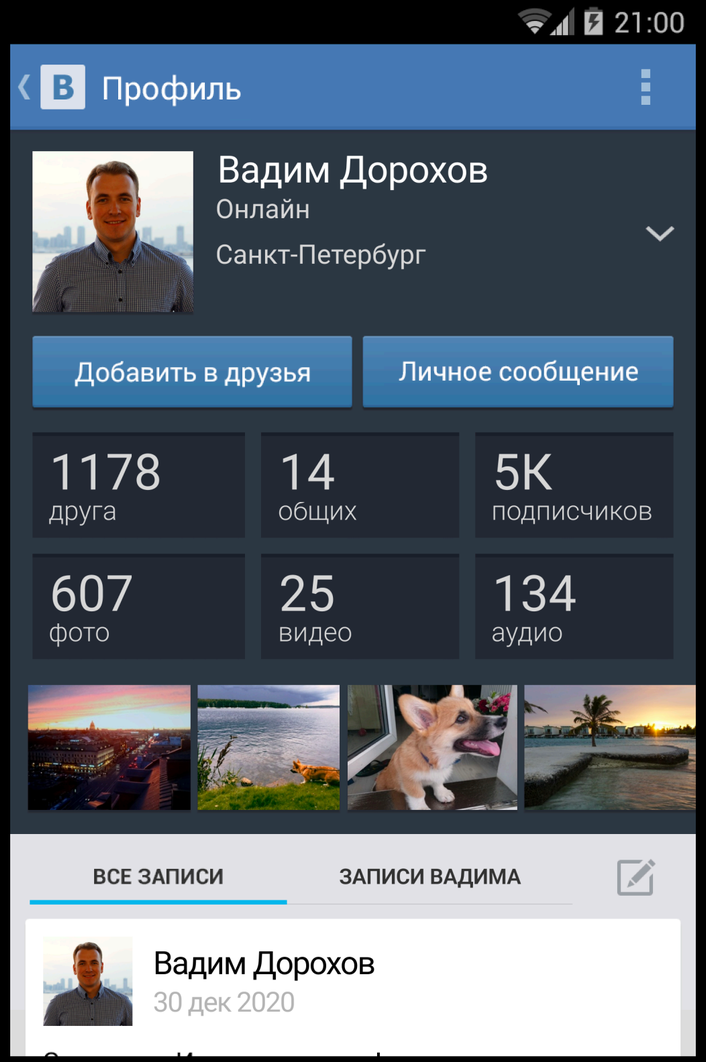
Тарас Иванов

- 4 года занимаюсь инфраструктурой
- Вместе улучшаем перформанс

Мобильная версия ВКонтакте

Как всё начиналось

- 2011 год: появилась мобильная версия для WAP-браузеров
- 2017 год: я второй разработчик на весь m.vk.com



- Супермедленный интернет
- Кнопочные телефоны
- Должна работать на микроволновке

Как сейчас

40 млн MAU 6 млн DAU

- Mobile First SPA
- Единый бэкенд
- 92% пользователей поддерживают ES2021

- PWA-версия
- Развивается 10+ продуктовыми командами
- Не уступает нативным клиентам по фичам

Платформа росла быстрее, чем оптимизировалась

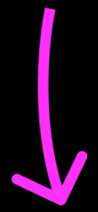
Почему производительность важна

- Гиганты индустрии заявляют, что скорость открытия влияет на бизнес-метрики
- И она действительно влияет!

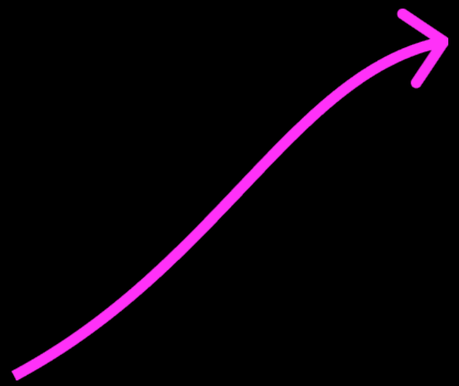
FCP, 75-й перцентиль

О чём поговорим

Какие существуют метрики производительности



Как мы можем на них повлиять?



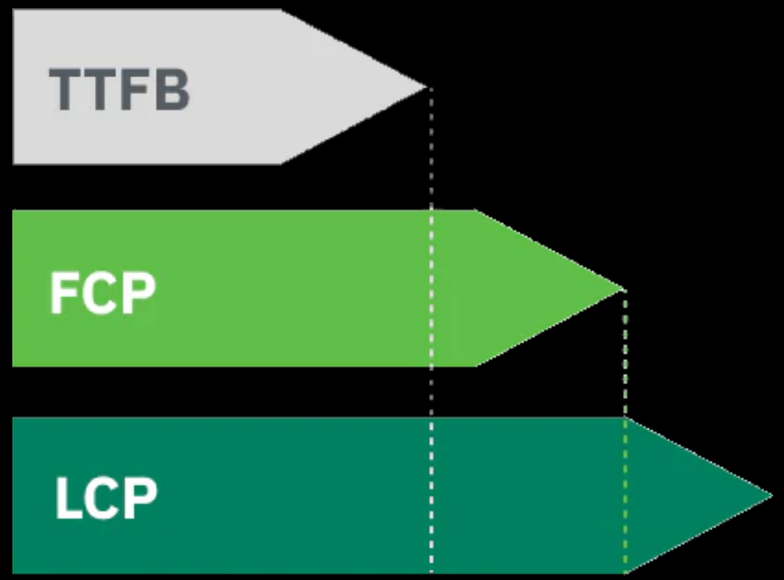
Принципиальная схема ускорения рендера в любом проекте

Как это всё влияет на продуктовые показатели?

Чтобы что-то сделать лучше,
надо понять, что именно

Концепция Web Vitals

Web Vitals



TTFB — время до получения первого байта (вычисления на сервере + сеть)

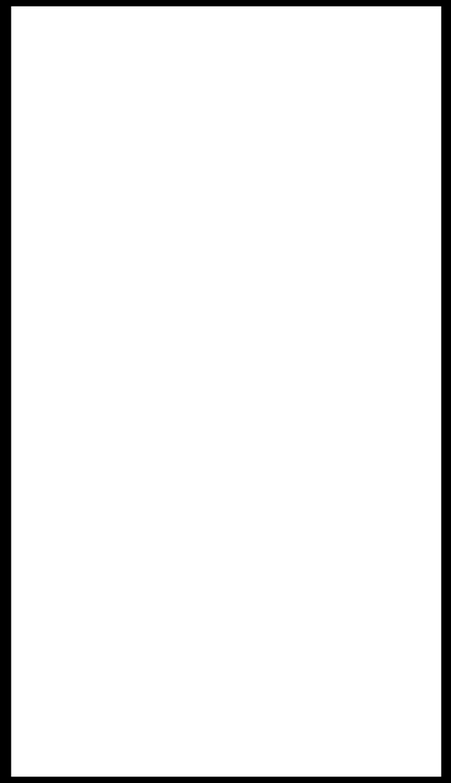
FCP — время до того, как на экране появится хоть что-нибудь

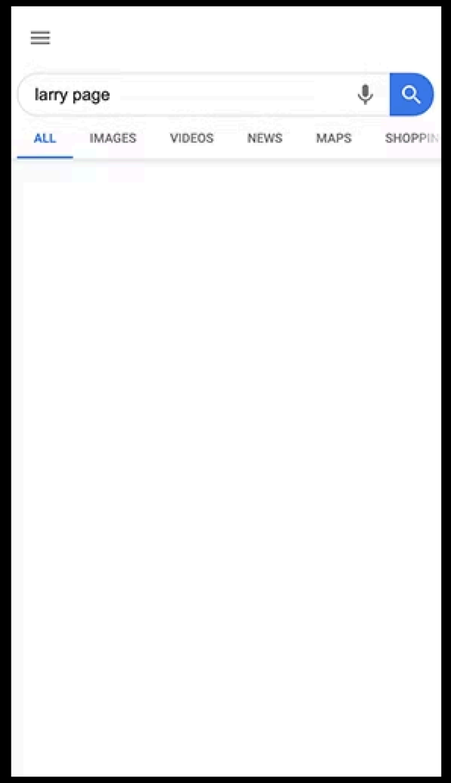
LCP — время до того, как на экране появится полезный контент

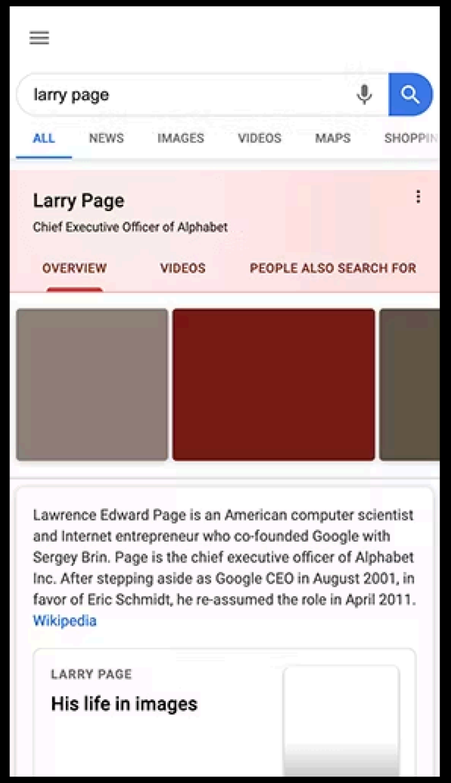
TTFB

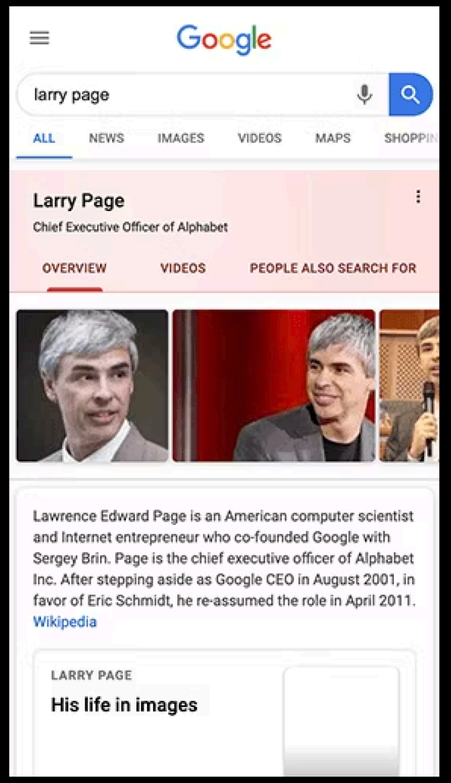
FCP

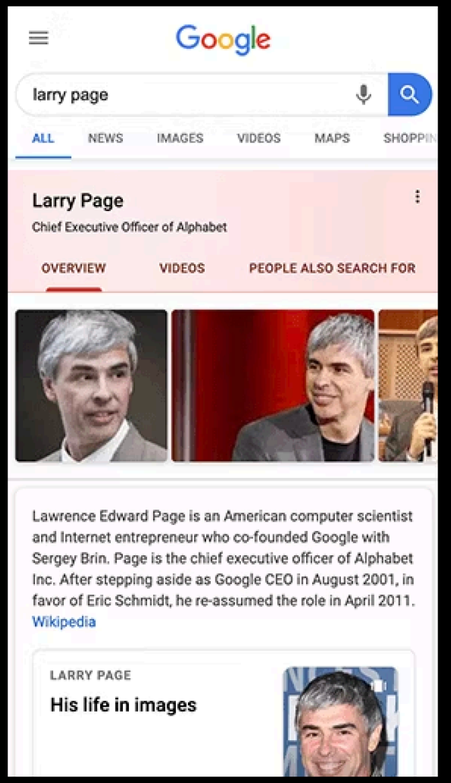
LCP




larry page
ALL
IMAGES
VIDEOS
NEWS
MAPS


larry page
ALL
NEWS
IMAGES
VIDEOS
MAPS
Larry Page
Chief Executive Officer of Alphabet
OVERVIEW
VIDEOS
PEOPLE ALSO SEARCH FOR
Lawrence Edward Page is an American computer scientist and Internet entrepreneur who co-founded Google with Sergey Brin. Page is the chief executive officer of Alphabet Inc. After stepping aside as Google CEO in August 2001, in favor of Eric Schmidt, he re-assumed the role in April 2011.
Wikipedia
LARRY PAGE
His life in images


Google
larry page
ALL
NEWS
IMAGES
VIDEOS
MAPS
Larry Page
Chief Executive Officer of Alphabet
OVERVIEW
VIDEOS
PEOPLE ALSO SEARCH FOR
Lawrence Edward Page is an American computer scientist and Internet entrepreneur who co-founded Google with Sergey Brin. Page is the chief executive officer of Alphabet Inc. After stepping aside as Google CEO in August 2001, in favor of Eric Schmidt, he re-assumed the role in April 2011.
Wikipedia
LARRY PAGE
His life in images


Google
larry page
ALL
NEWS
IMAGES
VIDEOS
MAPS
Larry Page
Chief Executive Officer of Alphabet
OVERVIEW
VIDEOS
PEOPLE ALSO SEARCH FOR
Lawrence Edward Page is an American computer scientist and Internet entrepreneur who co-founded Google with Sergey Brin. Page is the chief executive officer of Alphabet Inc. After stepping aside as Google CEO in August 2001, in favor of Eric Schmidt, he re-assumed the role in April 2011.
Wikipedia
LARRY PAGE
His life in images

Измеряем Web Vitals

Как измерить

Сервисы:

- Lighthouse
- Performance (DevTools)

Из кода:

- npm-пакет web-vitals, обёртка над new PerformanceObserver

На что обратить внимание

- Lighthouse и Performance — только для дебага, не опирайтесь на эти метрики в проде!
- Важно собирать метрики с реальных пользователей

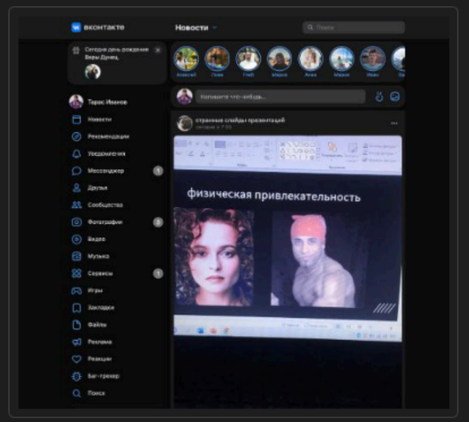
Локально

92

Performance

Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.

▲ 0–49 ■ 50–89 ● 90–100



METRICS Expand view

● First Contentful Paint
0.4 s

● Largest Contentful Paint
1.1 s

● Total Blocking Time
10 ms

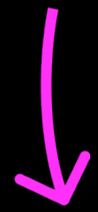
● Cumulative Layout Shift
0

На 75-м перцентиле

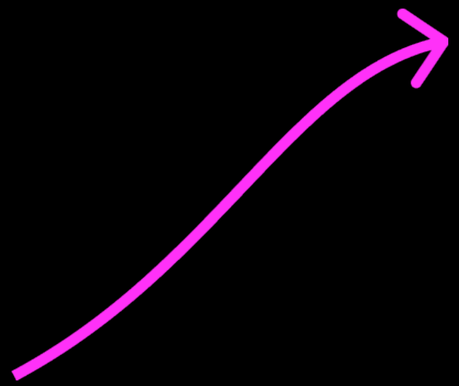
2.8 с.

О чём поговорим

Какие существуют метрики производительности



Как мы можем на них повлиять?



Принципиальная схема ускорения рендера в любом проекте

Как это всё влияет на продуктовые показатели?

Как мы можем улучшить Web Vitals

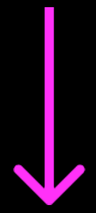
Как происходит отрисовка

Браузер парсит страницу по мере загрузки сверху вниз

Натыкается на блокирующие скрипты

↓

Скачивает их, исполняет и только потом идёт дальше

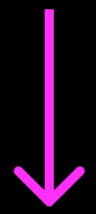


```
<html>
 <head>
   <title>Очень медленный сайт</title>

   <script src="https://cdn-1.vk.com/dist/polyfills.js"></script>
   <script src="https://cdn-2.vk.com/dist/common.js"></script>
   <script src="https://cdn-3.vk.com/dist/apps.js"></script>

   <link rel="stylesheet" href="https://cdn-500.vk.com/dist/common.css" type="text/css" />
   <link rel="stylesheet" href="https://cdn-501.vk.com/dist/apps.css" type="text/css" />
 </head>
 <body>
   <div class="Layout">
     <div class="Layout__header">
       <h1 class="Layout__title">Шапка сайта</h1>
```

Скачивает стили

↓

Строит CSS

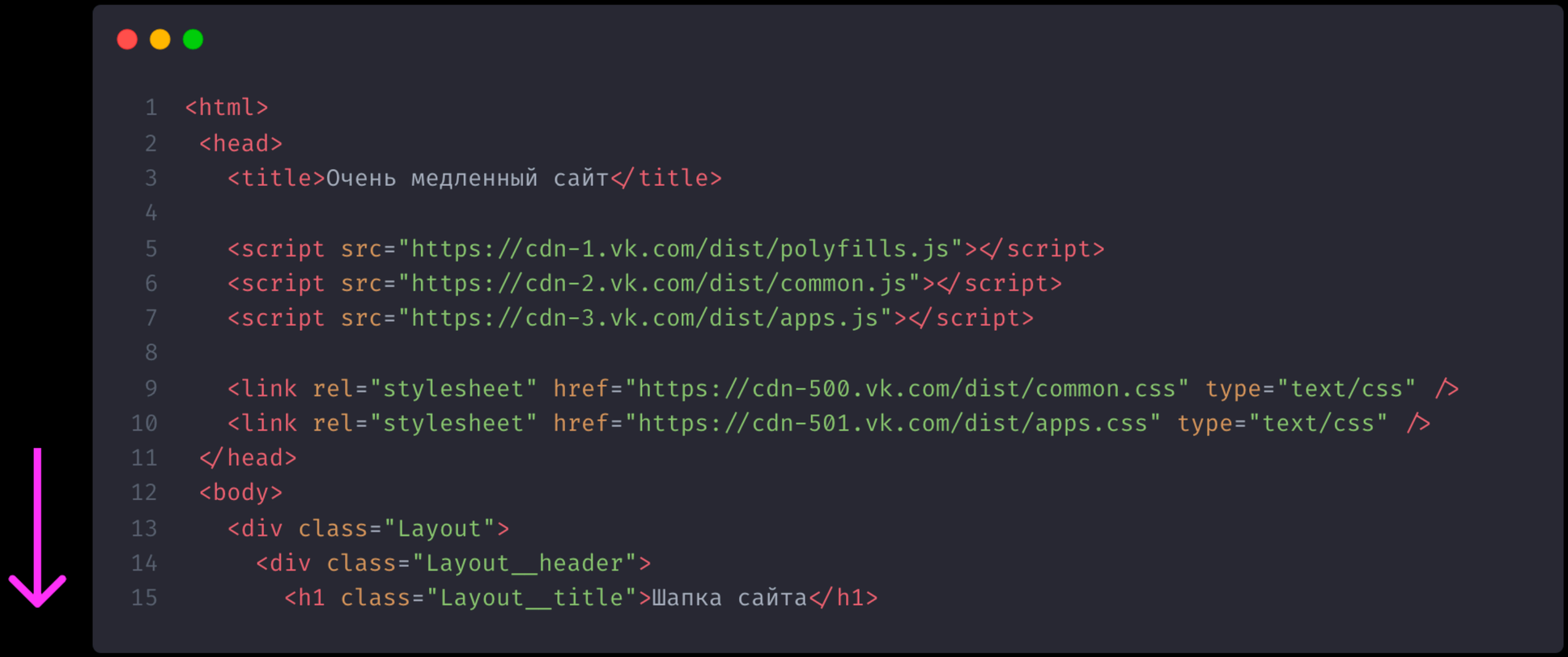


```html
<html>
 <head>
   <title>Очень медленный сайт</title>

   <script src="https://cdn-1.vk.com/dist/polyfills.js"></script>
   <script src="https://cdn-2.vk.com/dist/common.js"></script>
   <script src="https://cdn-3.vk.com/dist/apps.js"></script>

   <link rel="stylesheet" href="https://cdn-500.vk.com/dist/common.css" type="text/css" />
   <link rel="stylesheet" href="https://cdn-501.vk.com/dist/apps.css" type="text/css" />
 </head>
 <body>
   <div class="Layout">
     <div class="Layout__header">
       <h1 class="Layout__title">Шапка сайта</h1>
```

Скачивает весь HTML

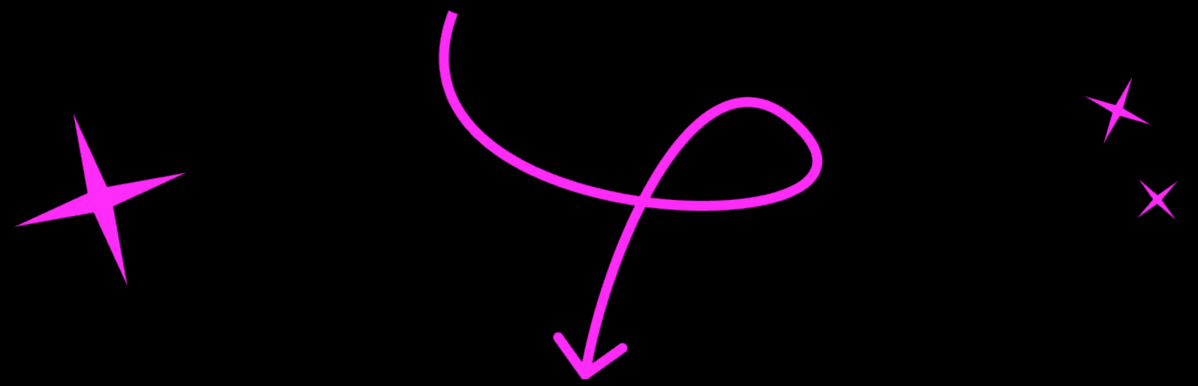
↓

Строит DOM



```
<html>
 <head>
   <title>Очень медленный сайт</title>

   <script src="https://cdn-1.vk.com/dist/polyfills.js"></script>
   <script src="https://cdn-2.vk.com/dist/common.js"></script>
   <script src="https://cdn-3.vk.com/dist/apps.js"></script>

   <link rel="stylesheet" href="https://cdn-500.vk.com/dist/common.css" type="text/css" />
   <link rel="stylesheet" href="https://cdn-501.vk.com/dist/apps.css" type="text/css" />
 </head>
 <body>
   <div class="Layout">
     <div class="Layout__header">
       <h1 class="Layout__title">Шапка сайта</h1>
```

Склеивает DOM + CSS в RenderTree



Немного магии — и готово!

Но это не так!

Ну... не совсем так

- Чтобы что-то отрисовать, браузеру не нужно целиком загружать страницу
- Достаточно распарсить часть HTML и часть CSS

А это значит, нужно...

- Убрать все блокирующие запросы
- Заинлайнить CSS, который нужен, чтобы отрисовать лэйаут
- Перенести верстку лэйаута в самый верх страницы
- Браузеру будет достаточно распарсить первый чанк HTML, чтобы сразу отрисовать страницу

Вот как это выглядит в теории

Defer 5–8

Preload 9–10

Inline CSS 12–16

Parse HTML 19–21

```
1  <html>
2    <head>
3      <title>Очень быстрый сайт</title>
4
5      <script src="https://cdn-1.vk.com/dist/polyfills.js" defer></script>
6      <script src="https://cdn-2.vk.com/dist/common.js" defer></script>
7      <script src="https://cdn-3.vk.com/dist/apps.js" defer></script>
8
9      <link rel="preload" href="https://cdn-500.vk.com/dist/common.css" as="style" />
10     <link rel="preload" href="https://cdn-501.vk.com/dist/apps.css" as="style" />
11
12     <style>
13       .Layout { background: #000; }
14       .Layout__header { display: flex; }
15       .Layout__title { color: #fff; font-size: 24px; }
16     </style>
17   </head>
18   <body>
19     <div class="Layout">
20       <div class="Layout__header">
21         <h1 class="Layout__title">Шапка сайта</h1>
```

Звучит несложно

Если вы оптимизируете лендинг

Реальный проект — это:

- Огромное количество легаси
- 10–30 блокирующих рендеринг JS
- 5–10 CSS-бандлов
- Если не повезёт, ещё и inline JS, который вызывается прямо из HTML
- Inline-обработчики событий на DOM-нодах

Что делать — понятно...
А как делать-то?

Я расскажу, как делали мы

Оптимизации применимы к любому проекту

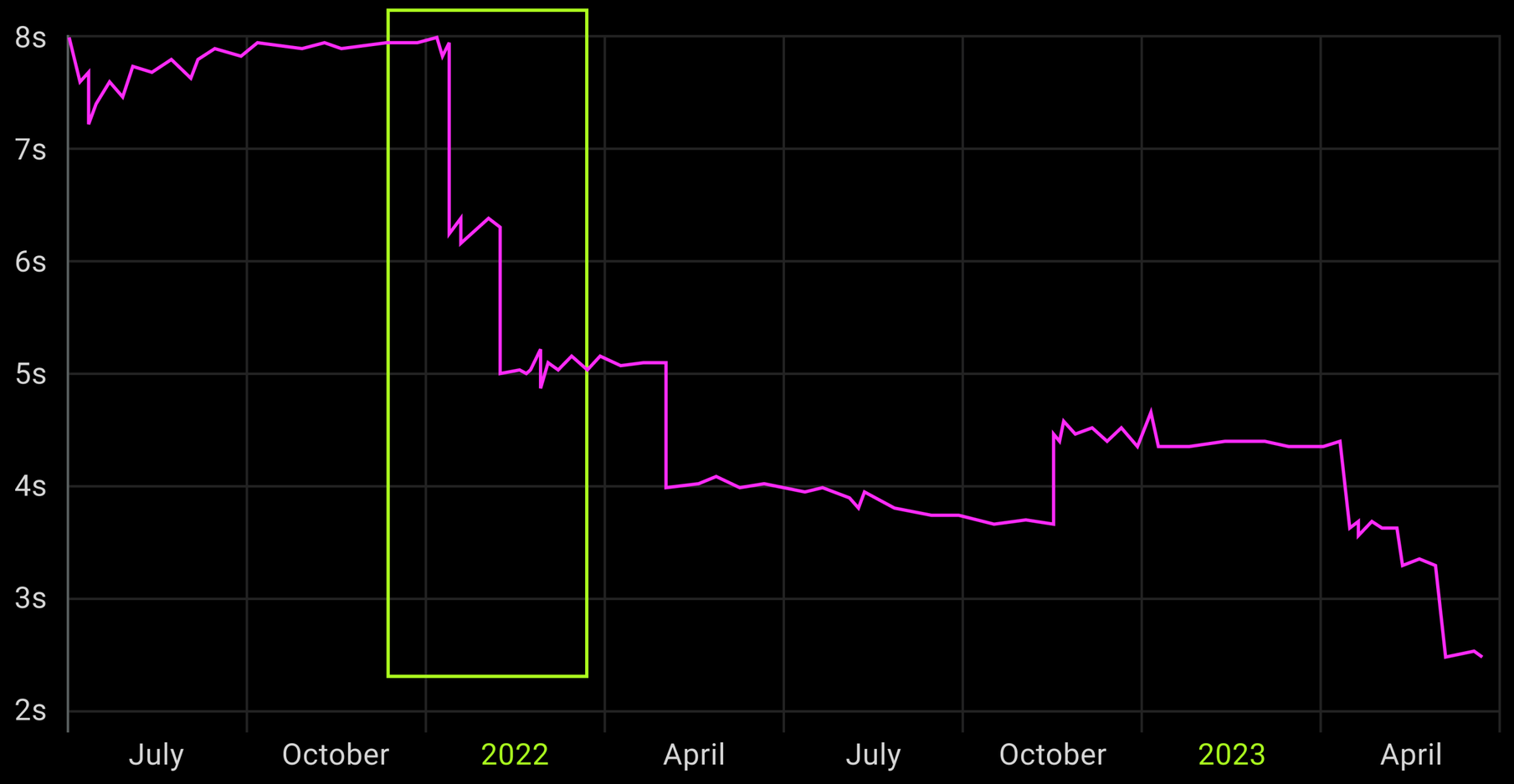
С чего начать?

шаг 1

Убираем всё лишнее

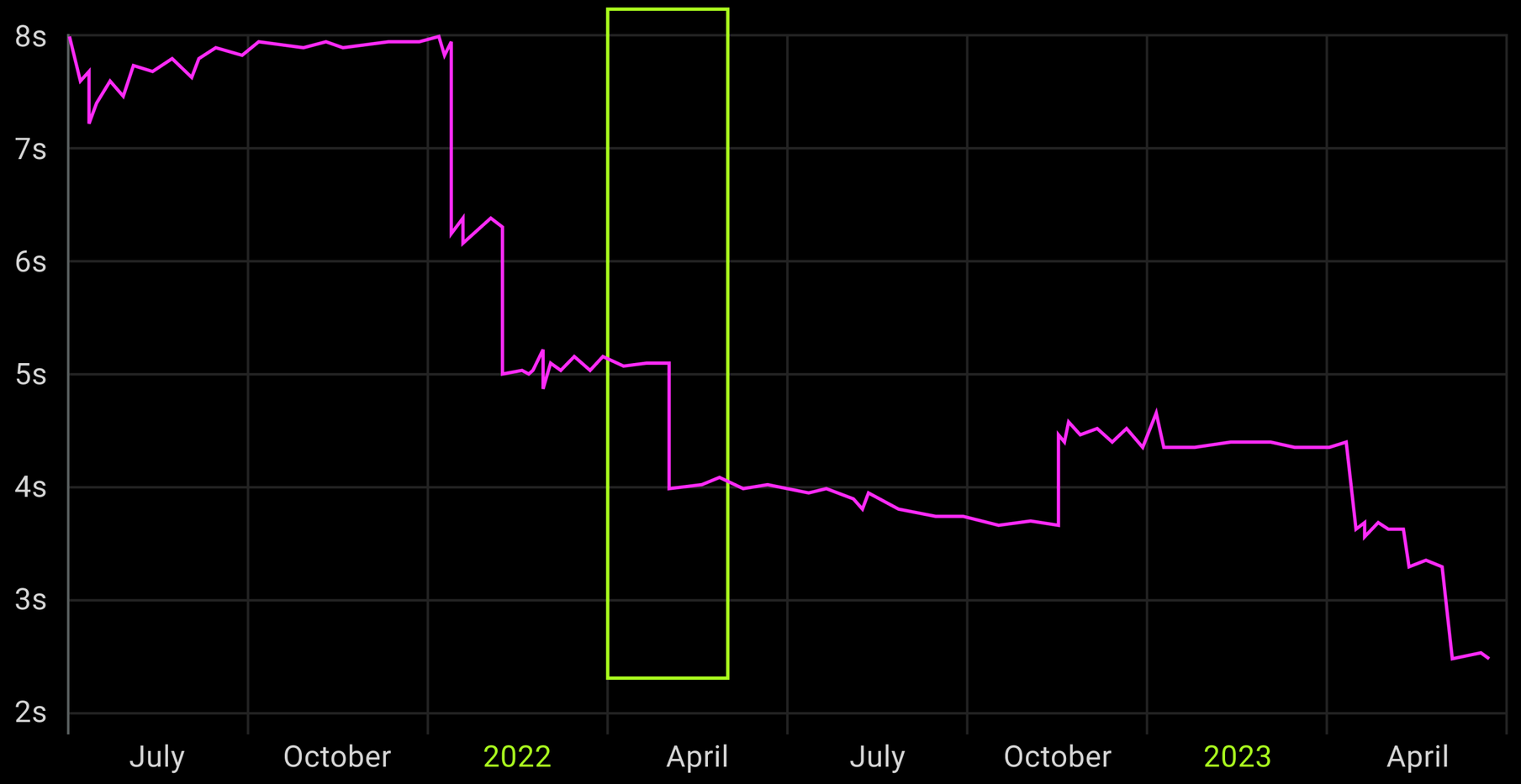
- Настраиваем сборщик
- Тришейкаем всё, что не нужно
- По возможности нарезаем код на бандлы и подключаем их по необходимости

Тришейкнули JS-бандлы на 2 Мб



FCP, 75-й перцентиль

Стали нарезать код на модули и вынесли дубли в отдельные чанки



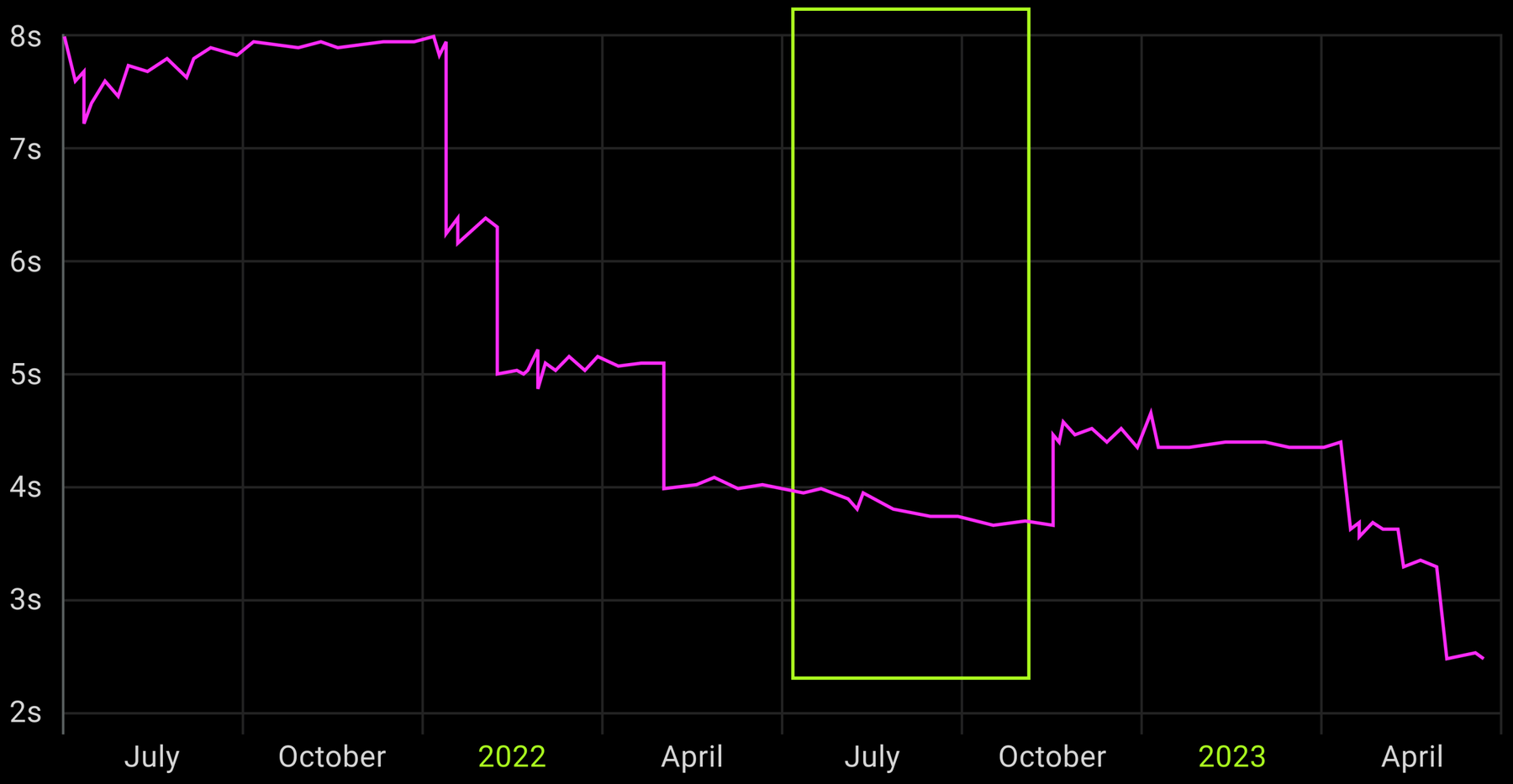
FCP, 75-й перцентиль

шаг 2

Early Hints

- Начинаем грузить CSS и JS до основного ответа страницы
- Используем серверные заголовки early-hints или dns-prefetch
- Подробнее об Early Hints на lightning talks

Поддержали
103 Early Hints



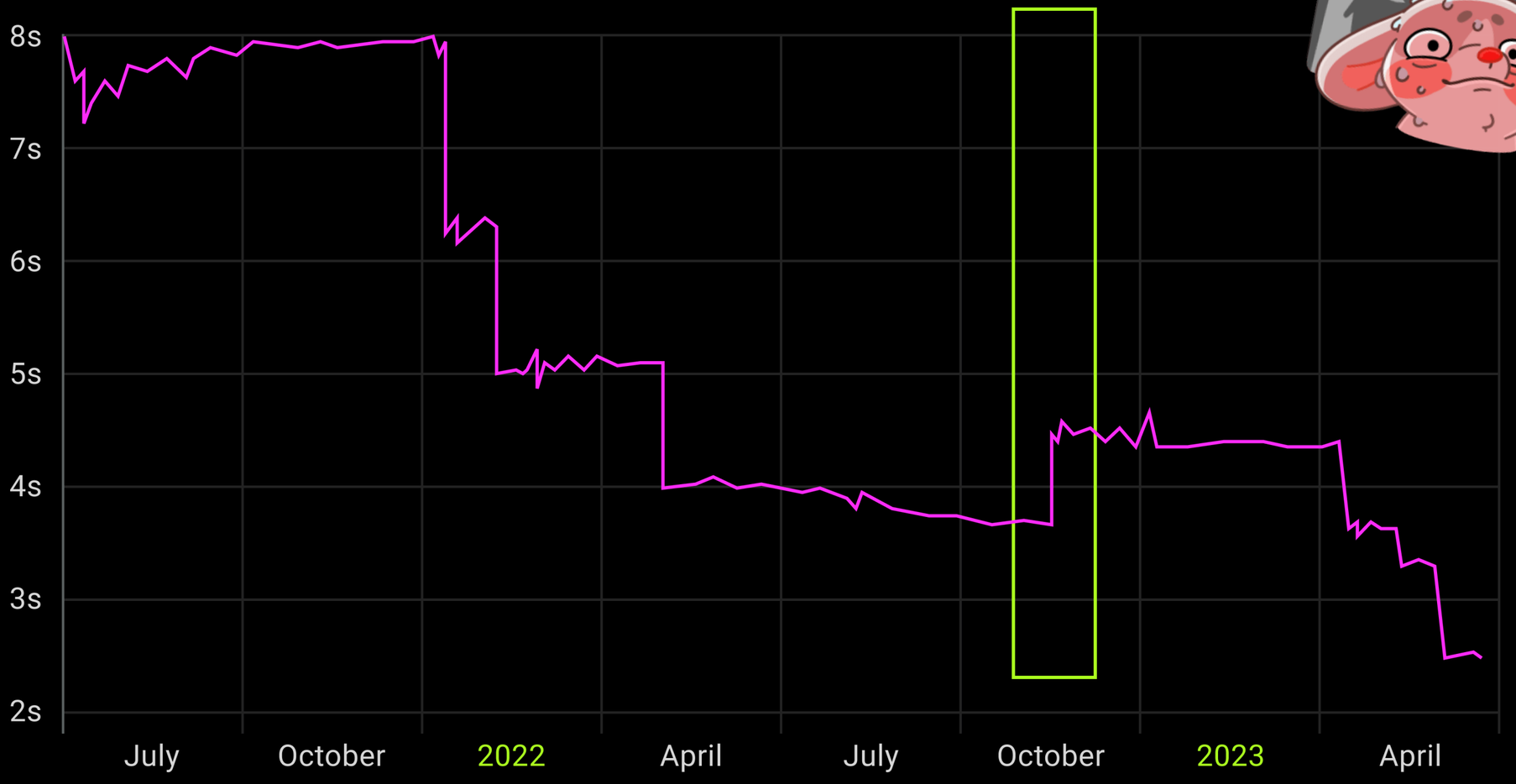
FCP, 75-й перцентиль

шаг 3

Не забываем про правильный сбор метрик

- Важно собирать метрики с реальных пользователей
- Чем больше вы их собираете, тем ближе к реальности ваша аналитика

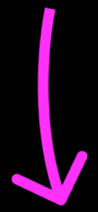
Стали собирать сильно больше Web Vitails



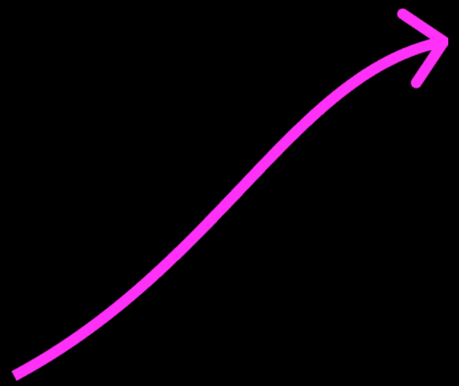
FCP, 75-й перцентиль

О чём поговорим

Какие существуют метрики производительности



Как мы можем на них повлиять?



Принципиальная схема ускорения рендера в любом проекте

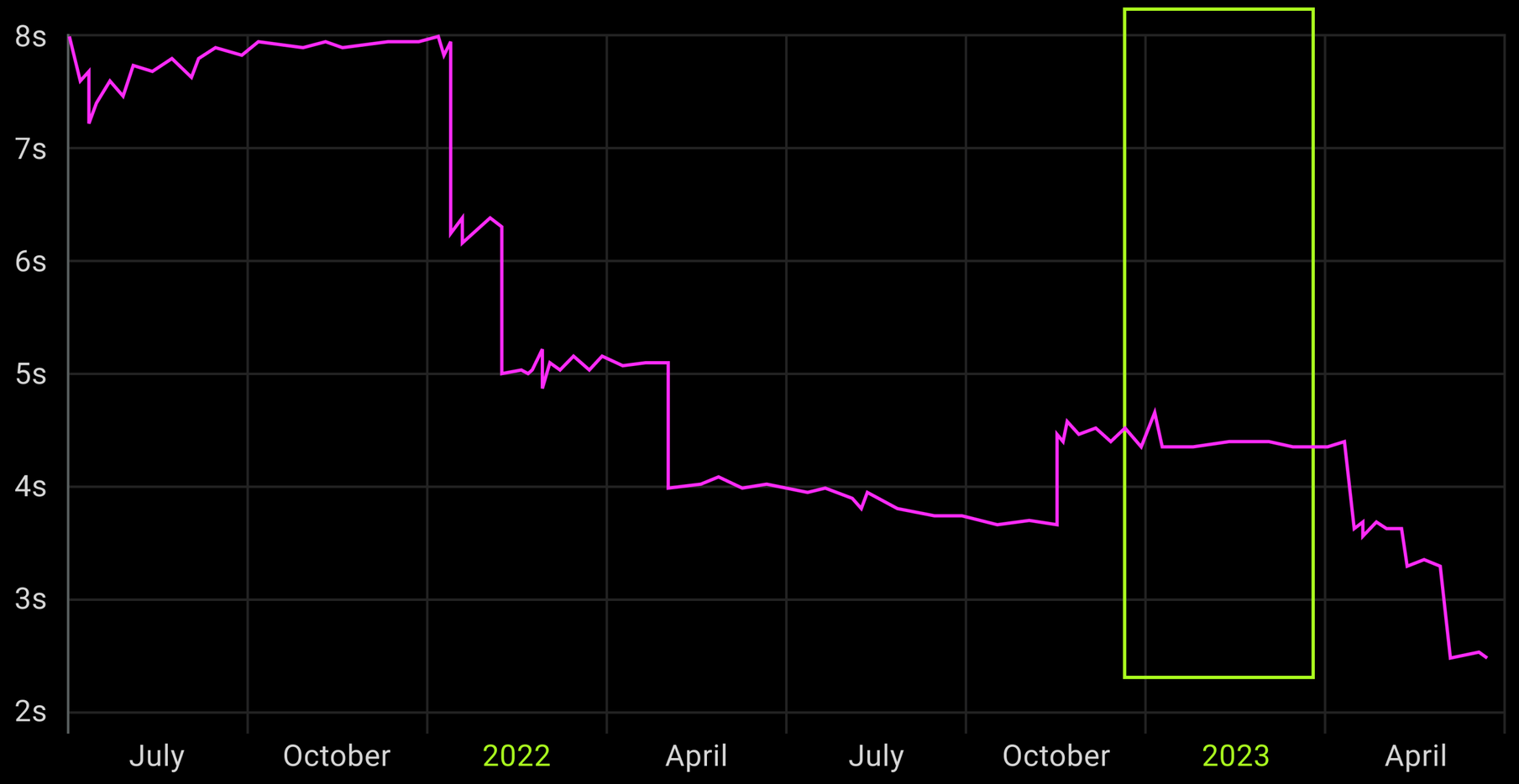
Как это всё влияет на продуктовые показатели?

шаг 4

То, ради чего мы тут собрались

- Механизм асинхронной загрузки статики StaticManager
- Генерация Inline CSS

Плыли во флэте,
пока пилили StaticManager



FCP, 75-й перцентиль

StaticManager

- Нужно научиться исполнять инлайн-код
после загрузки всех JS-файлов
- onLoad-событие не подходит
- Нужен свой механизм для отслеживания готовности
страницы StaticManager

StaticManager

При запуске страницы мы знаем полный
список всех подключаемых файлов

```
staticManager.files = [
    { 'polyfills.js', resolved: false },
    { 'common.js', resolved: false },
    { 'apps.js', resolved: false },
];
```

В конце каждого из этих JS пишем его резолвер, например:
staticManager.setResolved(‘chunkname.js’);

```
/* ── common.js ── */
function(t,e,r){var n=r(29694);t.exports=function(t,e){return new(n(t))(0===e?0:e)}},70698:function(t,e,r){"use strict";var n=r(19409),o=r(26214),i=r(75548),a=r(78399),u=r(49588),s=r(94357),c=r(6365),f=s.Map,l=s.has,h=s.set,p=n([].push);t.exports=function(t){var e,r,n,s=u(this),v=a(s),d=[],y=new f,g=i(t)?function(t){return t}:o(t);for(e=0;e<v;e++)n=g(r=s[e]),l(y,n)||h(y,n,r);return c(y,(function(t){p(d,t)})),d}},35969:function(t,e,r){"use strict";var n=r(39414),o=r(46632),i=r(78541),a=r(38402),u=r(95838),s=r(93355),c=r(91906),f=r(58678),l=r(90774),h=c("Promise"),p="AsyncFromSyncIterator",v=s.set,d=s.getterFor(p),y=function(t,e,r){var n=t.done;h.resolve(t.value).then((function(t){e(l(t,n))}),r)},g=function(t){t.type=p,v(this,t)};

staticManager.setResolved('common.js');
```

Код, который зависит от подключаемого JS, оборачиваем
в staticManager.onReady и добавляем в очередь на исполнение

```
staticManager.onReady(() => {
    /* some code */
});
```

После того как все файлы будут отмечены выполненными,
исполняем все подписки из очереди

Загрузка всех JS-скриптов стала отложенной,
кроме самого StaticManager.

Что, в свою очередь, ускоряет парсинг и отрисовку страницы

Что, если пользователь начнёт нажимать на все кнопки до того, как JS загрузился?

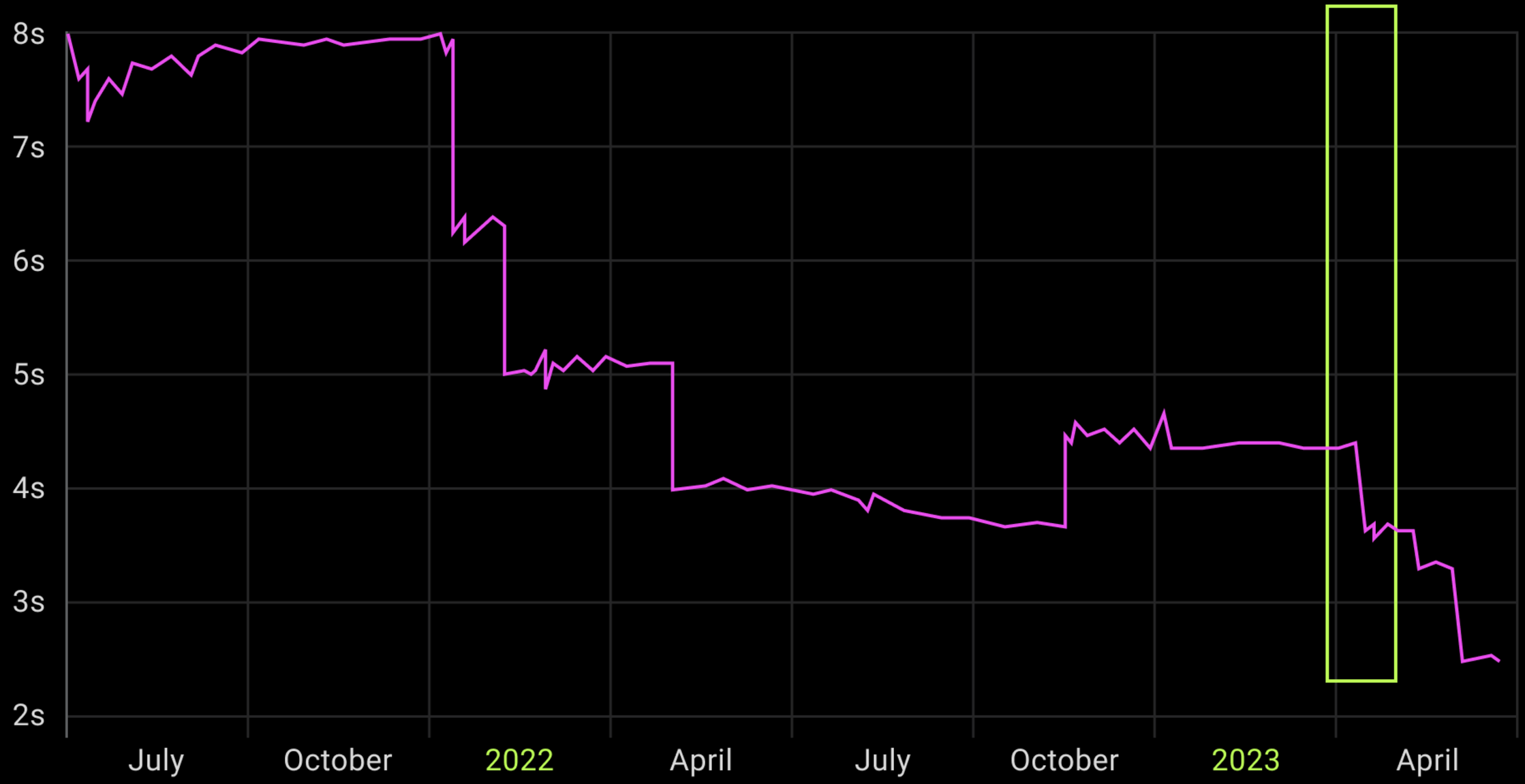
Proxy

- Мы можем проксировать все события,
которые генерирует пользователь
- Сложить их в очередь
- Вызвать их только после staticManager.onReady
или дропнуть, если прошло слишком много времени

Результаты A/Б

feed_fcp_mvk_q95	↓ -21.72%
feed_fcp_mvk_q75	↓ -23.23%
profile_fcp_mvk_q75	↓ -27.9%
video_fcp_mvk_q75	↓ -29.1%

FCP, 75-й перцентиль



Critical Path

- Рендерить страницу целиком,
  не скачивая CSS-бандлы
- Генерировать Inline CSS независимо
  от содержимого страницы

Как так сделать?

- Получаем список всех CSS-правил для всего проекта
- Когда HTML полностью построен, разбираем его на токены
- Классы, ID, теги
- Пересекаем с CSS-правилами

Inline CSS. Что на выходе

- Список всех необходимых и достаточных CSS-правил для рендера страницы независимо от динамически формируемого содержимого
- Засовываем его в шапку страницы
- Допиливаем StaticManager, чтобы он дожидался загрузки всех асинхронных CSS-бандлов и только потом вызывал staticManager.onReady

График запуска Inline CSS

Результаты А/Б

feed_ttfb_mvk_q75	↑ 2.35%
feed_lcp_mvk_q75	↓ -3.55%
feed_fcp_mvk_q75	↓ -5.5%

FCP, 75-й перцентиль

Что получилось?

- Мы не ждём загрузки JS-бандлов
- Мы не ждём загрузки CSS-бандлов
- Единственный блокирующий ресурс — StaticManager

Самая мякотка

Этого недостаточно

StaticManager блокирует рендер.
Мы хотим полностью избавится от таких ассетов

Решение

Инлайним код StaticManager и Proxy в страницу

Шучу

Или не шучу

- Вешаем на StaticManager.js атрибут defer,
отпустив браузер парсить дальше
- Пишем небольшой inline-скрипт, который только
собирает подписки на staticManager.onReady
- А после загрузки StaticManager.js передаёт их ему

А что насчёт пользовательских событий?

- Вешаем на все активные элементы
  pointer-events: none;
- Время такой блокировки будет несущественно маленьким и незаметным для пользователя

Профит

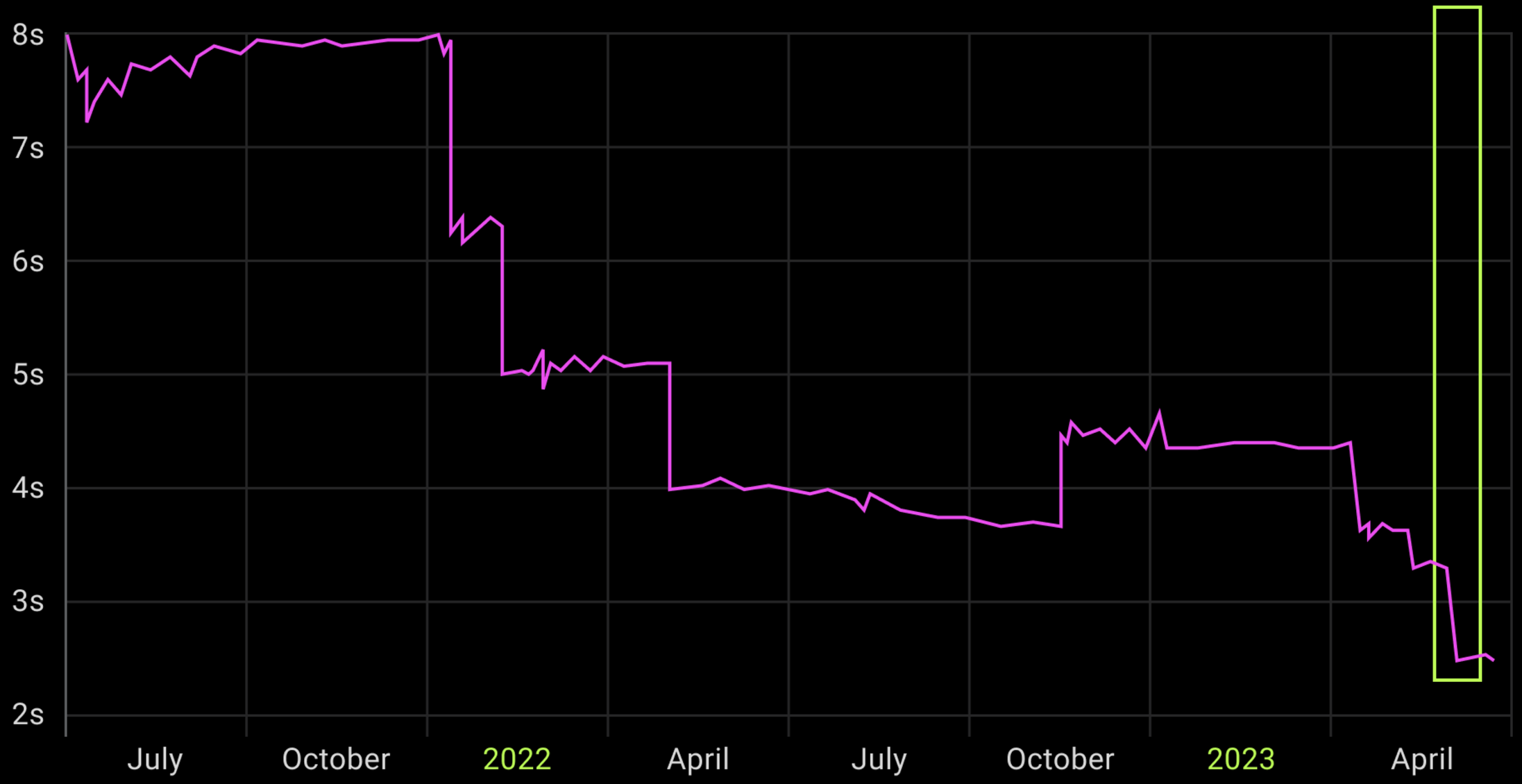
- Браузер сразу переходит к разбору HTML
- CSS строится только из используемых на странице стилей
- Как только первый кусок HTML будет разобран,
  он сразу будет отрисован
- Параллельно начинается загрузка изображений с CDN
- Эвристики браузера сами определяют, что нужно
  скачать в первую очередь, чтобы ускорить LCP
- Пользователь уже может взаимодействовать со
  страницей до полной загрузки

Мы полностью отказались от блокирующих ресурсов

Результаты A/Б

feed_lcp_mvk_q75	↓ -10.5%
feed_fcp_mvk_q75	↓ -14.3%
mail_lcp_mvk_q75	↓ -11.5%
mail_fcp_mvk_q75	↓ -14.8%
profile_fcp_mvk_q75	↓ -15.1%

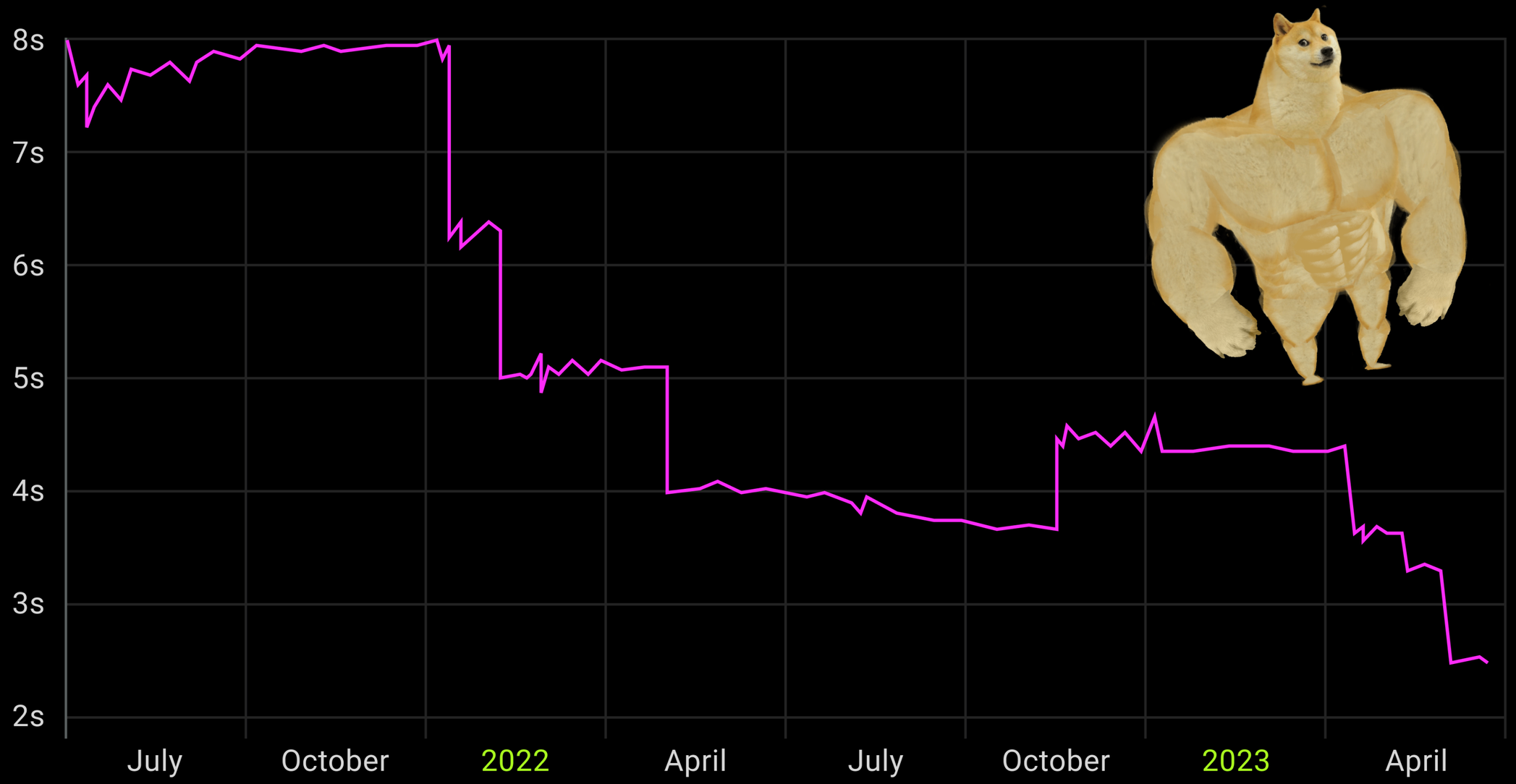
FCP, 75-й перцентиль



Ещё раз сравним до и после

FCP, 75-й перцентиль

FCP, 75-й перцентиль



Среднее FCP по всем разделам на q75

до: 8 с.

после: 2.5 с.

Как это помогло реальным пользователям?

Влияние на продукт

- Time spent: +1.2%
- Количество пользователей: +0.9%
- Прослушивание музыки: +20%
- Просмотры ленты: +3%
- Переходы в раздел приложений: +4.4%

Попробуйте у себя
на проекте прямо сейчас

Как узнать, что это даст вам

- Используйте WebPageTest
- Он позволяет провести эксперименты, не модифицируя код
- Прямо сейчас повесьте defer на все скрипты и посмотрите, насколько изменится время отрисовки

Если получилось у нас, получится и у вас

Материалы

- Мобильная версия: https://m.vk.com
- Про WebVitals: https://web.dev/vitals/
- npm web-vitals: https://www.npmjs.com/package/web-vitals
- PageSpeed: https://pagespeed.web.dev/
- WebPageTest: https://www.webpagetest.org/