Performance

Что такое производительность web-приложения?

К производительности веб приложений относят как набор измеримых показателей, так и воспринимаемый пользовательский опыт. Следующие области непосредственно влияют на наше восприятие производительности:

**Скорость загрузки.** На скорость загрузки влияет объем и количество ресурсов, которые необходимо загрузить для отображения нашего сайта или приложения. Чем меньше ресурсов и чем меньше их объем, тем быстрее пользователи смогут пользоваться нашим сайтом

**Первое взаимодействие.** Данная метрика подразумевает, что ресурсы не позволяющие пользоваться сайтом должны загружаться с наивысшим приоритетом, остальные ресурсы можно подгрузить в фоне. Ресурсы, которые блокируют пользовательское взаимодействие, называются блокирующими. Чем из меньше, чем меньше их размер, тем быстрее пользователи смогут взаимодействовать с нашим сайтом

**Плавность и интерактивность.** Помимо выше указанных характеристик на пользовательский опыт влияют такие понятия как плавность и интерактивность интерфейса. Грамотно подобные анимации. Плавный скролл контента создают у пользователя ощущение, что все работает как надо

**Воспринимаемая производительность.** На самом деле это даже не метрика, а некоторый набор уловок позволяющий оставлять пользователя вовлеченным в процесс взаимодействия с нашим сайтом, даже пока сайт не доступен. Этот показатель довольно субъективен и может отличаться для разных пользователей, но зачастую он не менее важен чем предыдущие характеристики.

Зачем нам нужно заботиться о производительности?

Если мы работаем на топовой модели Mac, наше приложение используется во внутренней корпоративной сети возможно производительность для нас не является чем-то первостепенным.

Раньше основными потребителями интернет трафика были компьютеры, но с недавних пор ситуация начала меняться. Уже сейчас на мобильные устройства приходится 60% всего интернет трафика, тогда как на планшеты и ПК суммарно приходится 39%. Более 25% контент Instagram, TikTok, YouTube и других медиа платформ просматривается с телефонов. Ежегодный рост поисковых запросов с мобильных устройств составляет более 5%. К сожалению, не у всех пользователей есть доступ к безлимитному Wi-Fi c последнего IPhone, подавляющее большинство пользователей выходит в интернет с дешевых Android из сети 3G.

В какой-то момент компании обнаружили что чем быстрее пользователь получит доступ к их сайту или web-приложению, тем выше вероятность что он вернется назад, воспользуется услугами компании или что-то купит.

Например, BBC выяснила, что каждая дополнительная секунда загрузки страницы отсеивала дополнительные 10% пользователей сайта. Доход BBC от рекламы за 2022 составил 306 000 000 £

Pinterest сократив время загрузки страниц на 40% увеличил на 15% число регистраций и переходов из поисковых систем. В 2021 всего 2.1 миллиарда долларов доходов от рекламы с перспективой роста до 4.1 миллиарда к 2026.

NCC Group сократив время загрузки страниц на 850мс увеличила конверсию на 7%, сократила число отказов на 7% и увеличила на 10% количество просмотренных за сеанс страниц.

Mobify увеличила годовой доход на 910 000 $ сократив время загрузки страниц на 100мс.

Напрашивается довольно простой вывод, хорошая производительность в web позволяет заработать.

Как мы можем улучшить нашу производительность?

Для достижения оптимальной производительности нам понадобится разобраться в таких вещах как Critical rendering path и Core web vitals. Первый является по сути алгоритмом, по которому обрисовываются наши страницы. Второе, это набор метрик, разработанных инженерами Google для понимания, что считать хорошей производительностью, а что нет.

Critical Rendering Path

Алгоритм, по которому браузер преобразует полученные HTML, CSS, JS в страницы, отображаемые на экране. Сначала браузер запрашивает HTML страницу. Полученную страницу браузер парсит превращая в DOM-дерево. DOM-дерево, это структура данных содержащая в себе всю информацию о контенте страницы. В процессе парсинга браузер может встречать ссылки на дополнительные ресурсы (стили, скрипты, изображения) инициируя их дозагрузку. Некоторые такие запросы (CSS, JS) являются блокирующими, до момента пока браузер не получит запрошенные ресурсы и не обработает их соответствующим образом. После того как браузер распарсил весь HTML и построил DOM, он приступает к построению CSSOM. CSSOM по сути является аналогом DOM, но содержит информацию о том как правильно стилизовать соответствующие элементы страницы. До построения CSSOM рендеринг не возможен, ввиду каскадной природы CSS (стиль элемента, наследуемый от родителя или от менее специфичного селектора, может быть переопределен), потому CSS относится к ресурсам, блокирующим рендеринг. Построив DOM и CSSOM браузер приступает к построению Render tree, которое включает в себя только видимый пользователю контент страницы. На основе render tree браузер рассчитывает размеры элементов страницы и их положение. Этот процесс называется Layout. После чего страница обрисовывается.

Core web vitals

Largest containtful paint

Метрика, показывающая время с момента начала загрузки страницы, до момента отрисовки наибольшего видимого целиком блока контента. Хорошим результатом считается отрисовка блока за 2,5с, все что до 4с считается приемлемым все что больше требует оптимизации.

На данный момент основными элементами, влияющими на расчет метрики являются изображения, включая изображения внутри svg элементов, видео, элементы с фоном загружаемым через url() и блочные элементы содержащие текст или инлайновые элементы с текстом.

Стоит отметить, что в процессе отрисовки наибольший видимый элемент может меняться. На метрику влияет последний наибольший элемент первого экрана, даже если он удаляется со страницы или скрывается стилем. Если элемент выходит за область видимости экрана, то в расчет попадает только его видимая часть.

Так же стоит отметить, что браузеры перестают рассчитывать метрику, после первого пользовательского взаимодействия со страницей.

На результаты расчетов LCP влияют несколько параметров:

* Медленное соединение
* Блокирующие ресурсы и время их загрузки
* Рендеринг приложения на клиенте

Для улучшения метрики прибегают к различным решениям. Стараются ускорить доставку ресурсов страницы, это может быть как ускорение текущего сервера, так и расположение ресурсов ближе к конечным пользователям. Оптимизируют скорость обнаружения ресурсов в процессе парсинга HTML путем приоритизации загрузки. В случае большого количества ресурсов, те из них, которые являются критическими стараются делать инлайновыми (загружающимися непосредственно с html), а остальные загружать асинхронно. В случае, если рендер блокируется исполнением скриптов, то стараются проводить декомпозицию задач, исполняемых скриптами. Стоит так же отметить, что для сайтов использующих client side rendering, LCP улучшается путем рендера контента на сервере.

First input delay

Метрика измеряет время с момента первого взаимодействия пользователя со страницей, до момента, когда браузер начинает обрабатывать эти взаимодействия. Считается хорошим тайминг до 100мс, приемлемым считается тайминг до 250мс, все что больше требует оптимизации.

Основными причинами задержки является загрузка и парсинг js или долгое выполнение js после загрузки.

Большие значения метрики приводят к недоступности таких элементов как <input>, <textarea>, <select>, <a> в независимости от того привязаны к ним обработчики или нет.

Данная метрика фокусируется на отзывчивости страницы в процессе загрузки, потому она фокусируется на дискретных событиях, таких как клики, тапы на экран и ввод с клавиатуры

Для улучшения метрики предлагается декомпозировать большие задачи (выполнение которых занимает более 50мс). Так же предлагается загружать в первую очередь ресурсы необходимые для отрисовки первых экранов. Стараться минимизировать каскадную загрузку ресурсов (один ресурс в процессе загрузки запускает загрузку другого ресурса итд.). Выносить в воркеры выполнение больших и ресурсоемких операций (например обработка изображений на canvas)

Cumulative layout shift

Метрика представляет собой меру наибольшего непредвиденного сдвига элементов страницы.

Сдвигом считается изменение положения элементов на странице между последовательными кадрами отрисовки.

На метрику влияет только изменение позиции существующих на странице элементов. Добавление новых элементов или изменение размеров, уже существующих не учитывается до тех пор, пока остальные элементы остаются на своих местах.

Расчет метрики основан на размерах окна браузера и размерах смещения элементов между двумя последовательными кадрами.

**layoutShiftScore = impactFraction \* distanceFraction**

**impactFraction** - отношение объединения областей занимаемых элементами до и после смещения, к размеру viewport

**distanceFraction** - измеряет величину смещения элементов относительно viewport. Является отношением максимального смещения одного из элементов к максимальному измерению viewport (width, height)

Наиболее частыми причинами возникновения сдвига layout являются:

*Изображения без размеро*в. Исправляется добавлением размеров или ratio

*Рекламные блоки и встраиваемые элементы*. Правится резервированием места под встраиваемый блок. Так же рекомендуется не размещать подобные блоки в самом верху страницы, надо контентом, который уже отрисован

*Загрузка web-fonts*. Догружаемые шрифты могут занимать больше места чем те, с которыми страница от рисовалась изначально, что может приводить изменению размеров текстовых областей

*Анимации*. Предлагается не использовать свойства, приводящие к перерасчету layout

Инструменты

[Chrome User Experience Report](https://developer.chrome.com/docs/crux/)

Набор данных, предоставляющий информацию о реальном пользовательском опыте с сайтов

[PageSpeed Insights](https://developers.google.com/speed/pagespeed/insights/)

Показывает агрегированные данные по производительности за последние 28 дней. Так же предоставляет рекомендации по улучшению метрик

[Search Console (Core Web Vitals report)](https://support.google.com/webmasters/answer/9205520)

Предоставляет данные о производительности постранично. Содержит информацию только о сайтах, которыми вы владеете

[Библиотека web-vitals](https://github.com/GoogleChrome/web-vitals)

Предоставляет API для сбора информации по ключевым метрикам Core Web Vitals

[Chrome Dev Tools (Performance)](https://developer.chrome.com/docs/devtools/)

Вкладка в DevTools предоставляющая возможности скрининга производительности страницы. Включают в себя таймлайн загрузки и выполнения ресурсов, а также инструменты анализа производительности исполнения кода

[Lighthouse](https://developer.chrome.com/docs/lighthouse/overview/)

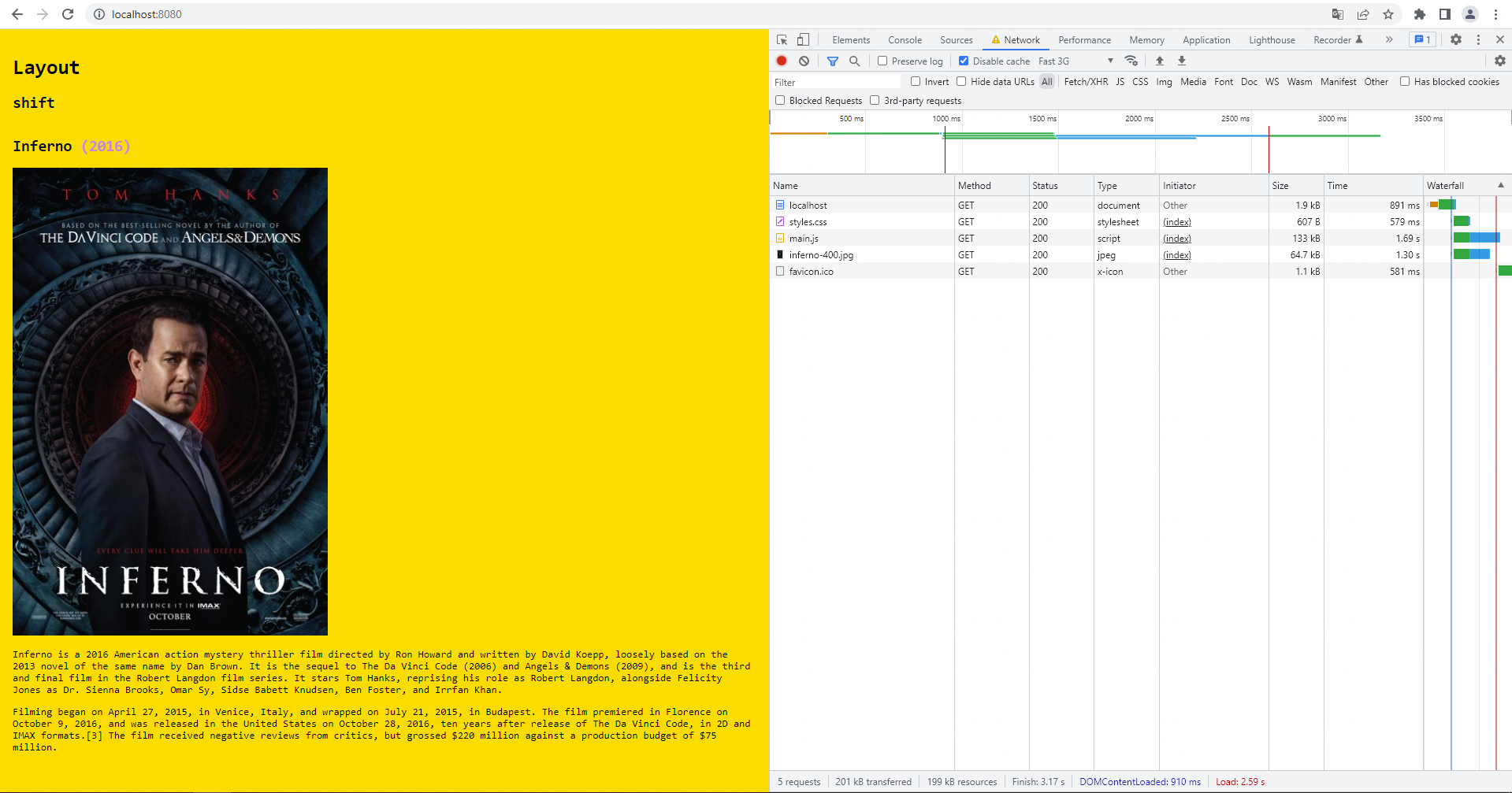
Утилита, собирающая в автоматическом режиме данные о метриках сайта и предоставляющая рекомендации о улучшении производительности

DevTools

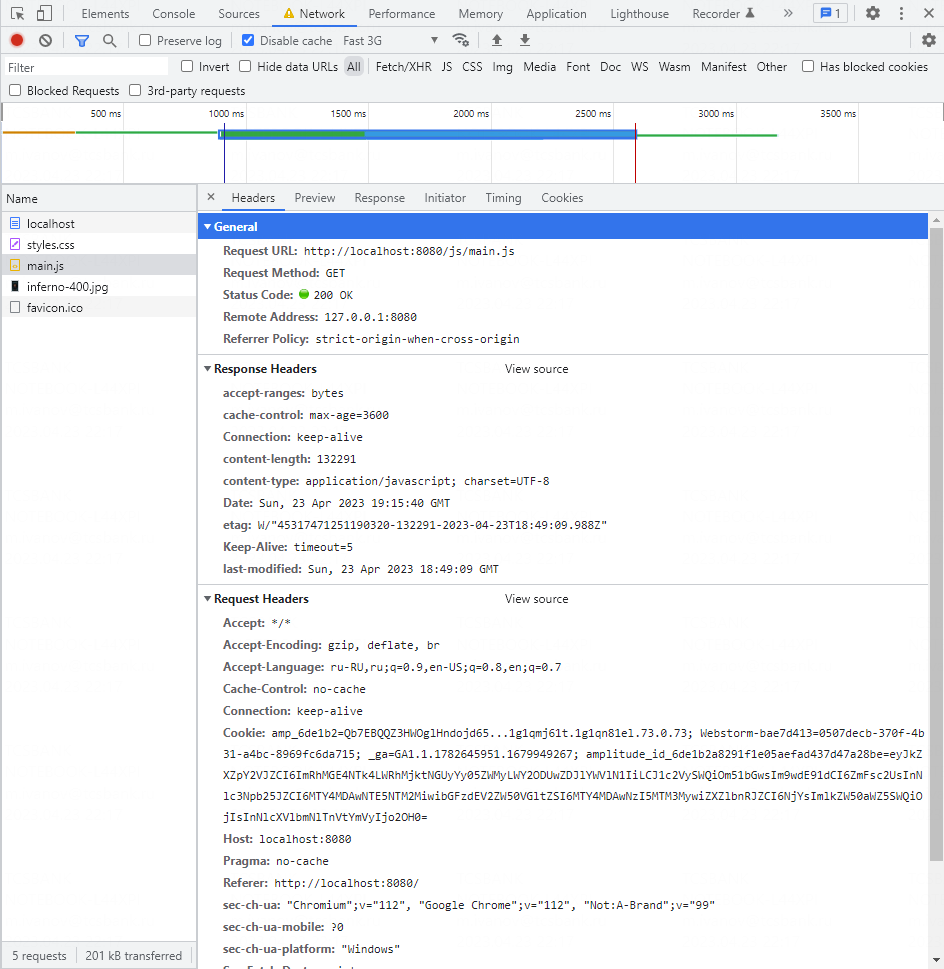
Network

Позволяет просматривать список загружаемых ресурсов.

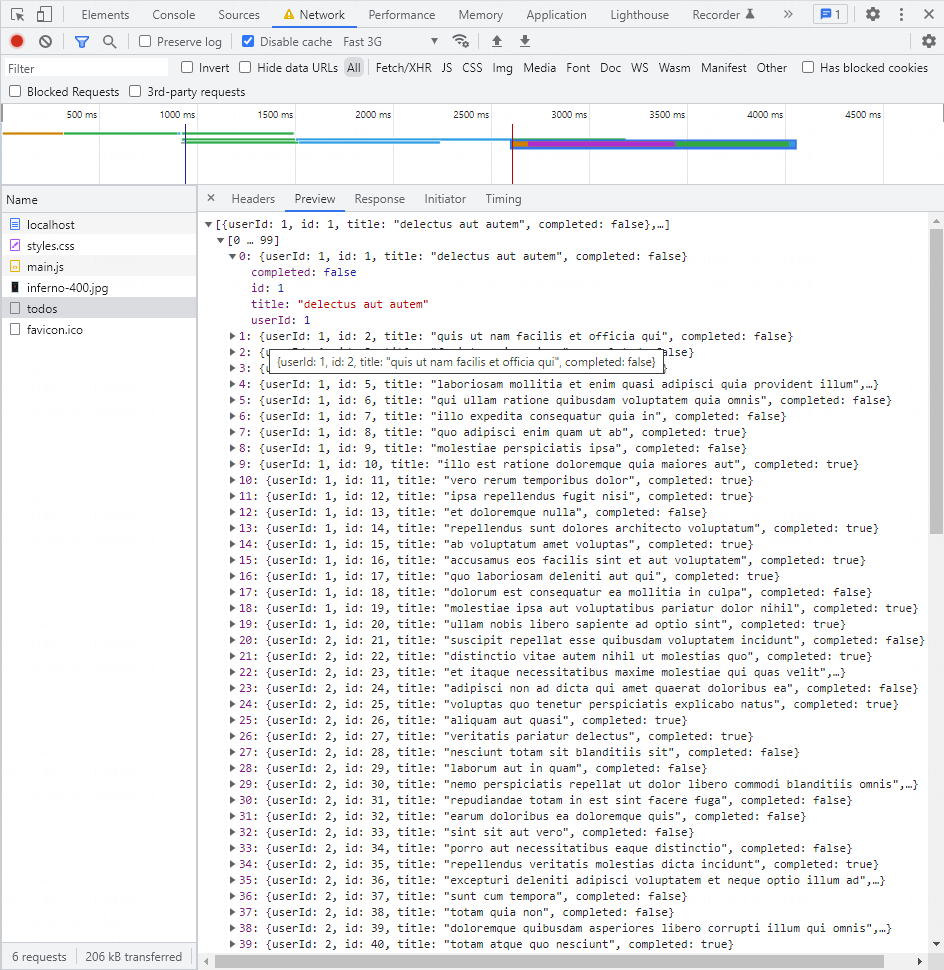
Содержит таймлайн загрузки.



При клике на ресурс из списка появляется панель просмотра информации о запросе.



Содержит вкладки просмотра заголовков запроса.



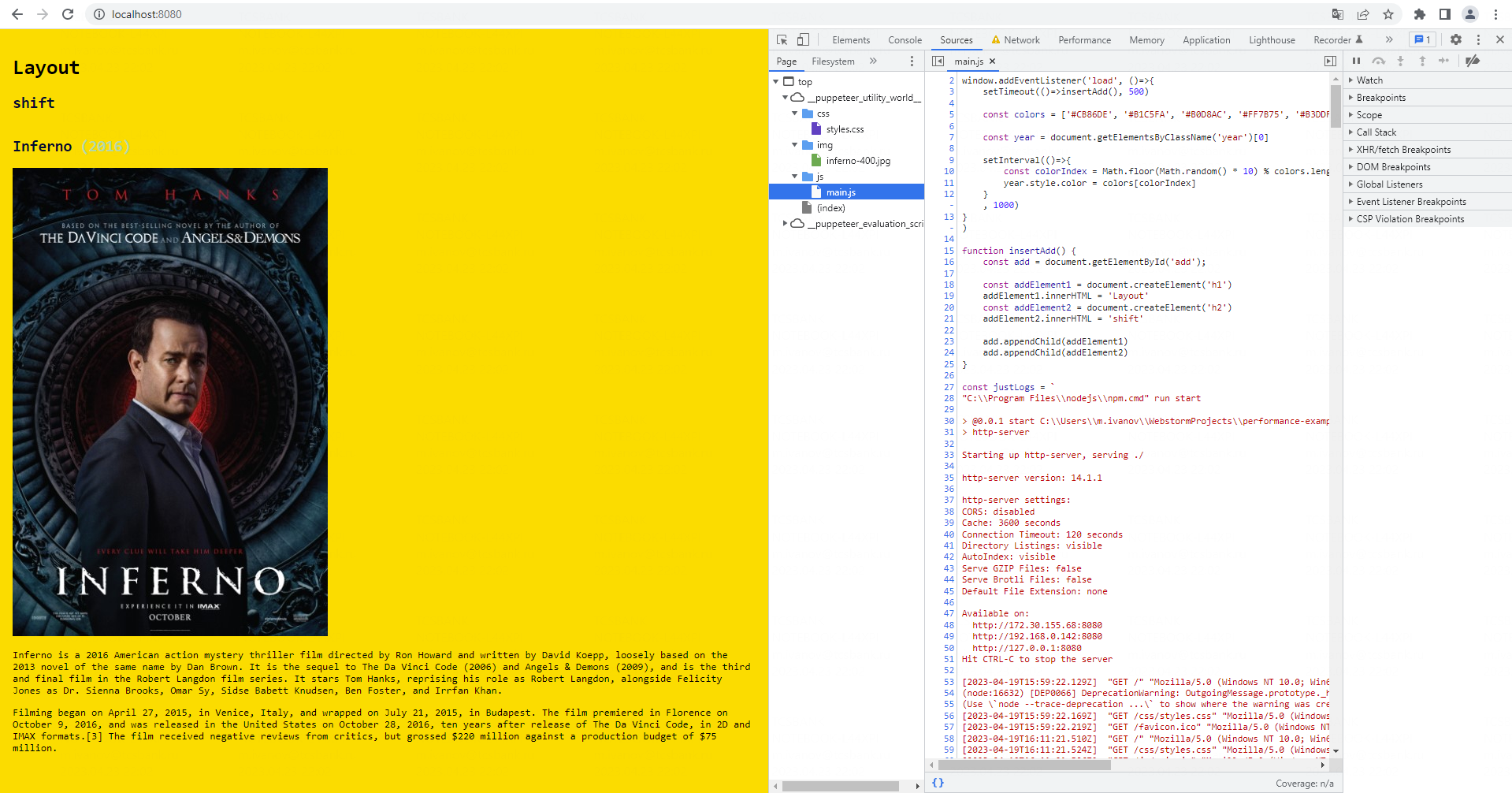
Предпросмотра ответа в сыром и отформатированном виде

А так же возможности просмотра инициатора запроса, таймингов и куки

Source

Предоставляет возможность просмотра ресурсов нашей страницы.

В левом области располагается панель просмотра ресурсов. Справа блок просмотра выбраного ресурса

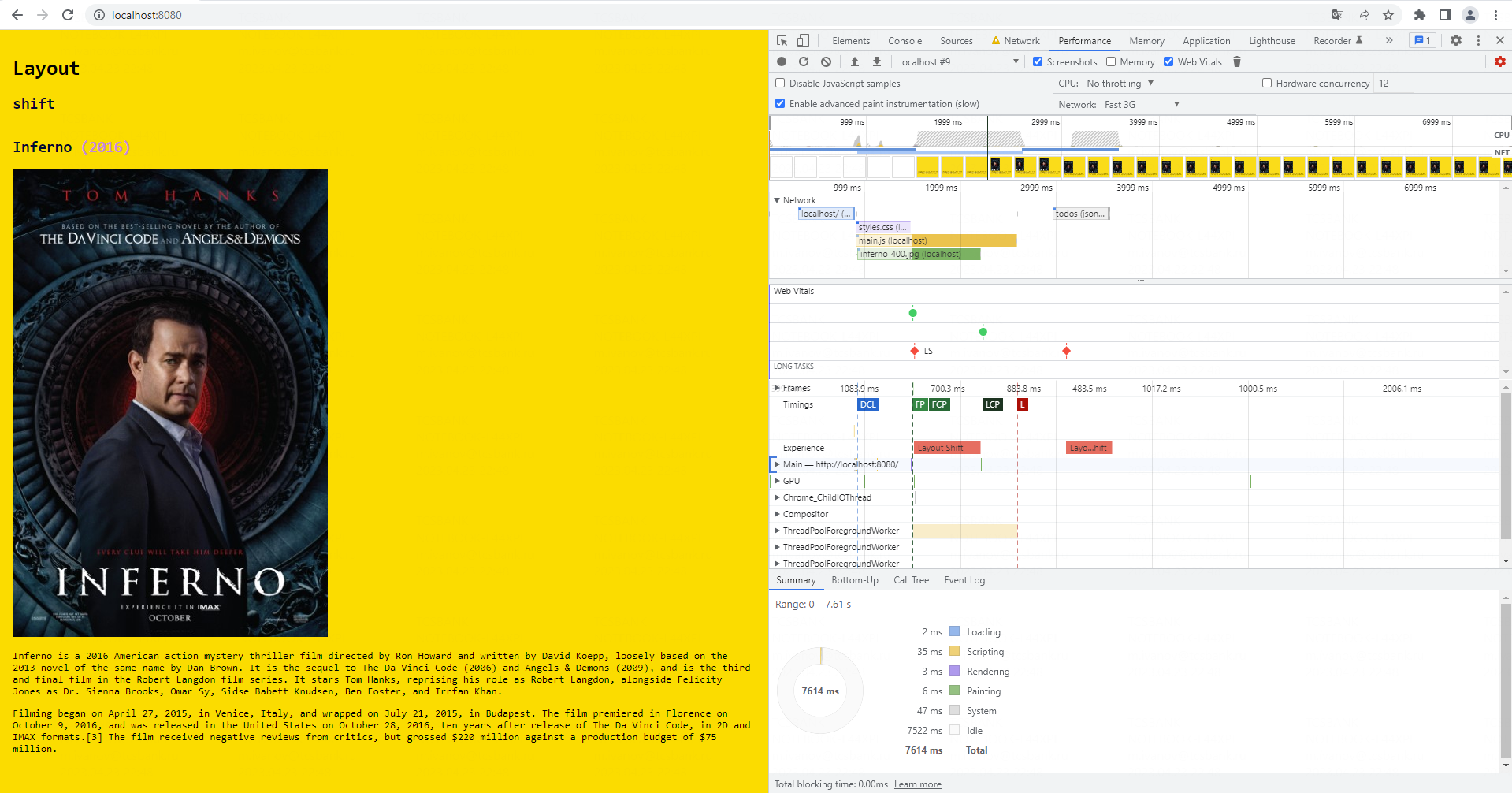


Performance

Содержит таймлайн с раскадровкой загрузки

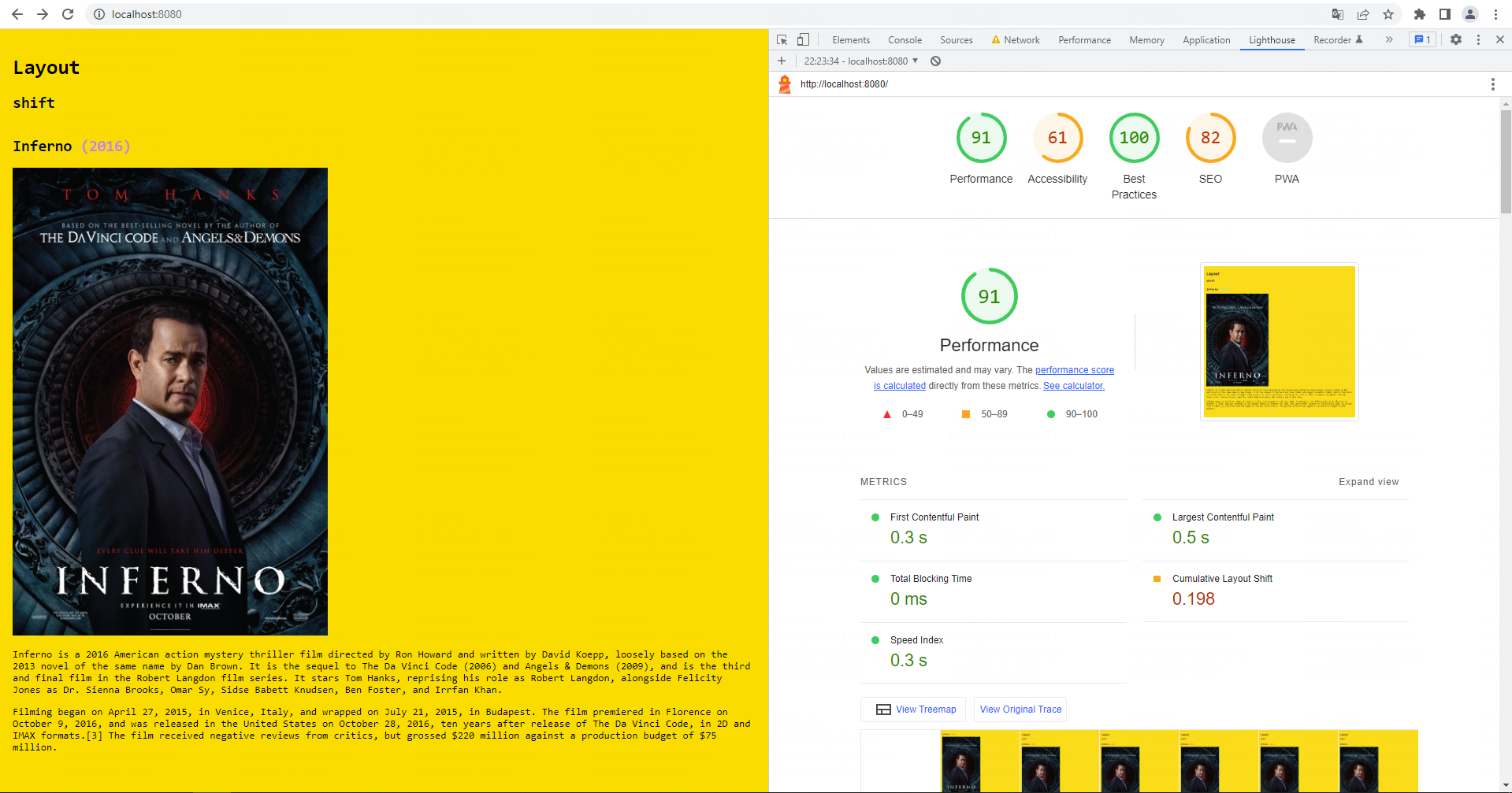
Диаграмму очередности загрузки ресурсов

Таймлайн veb vitals

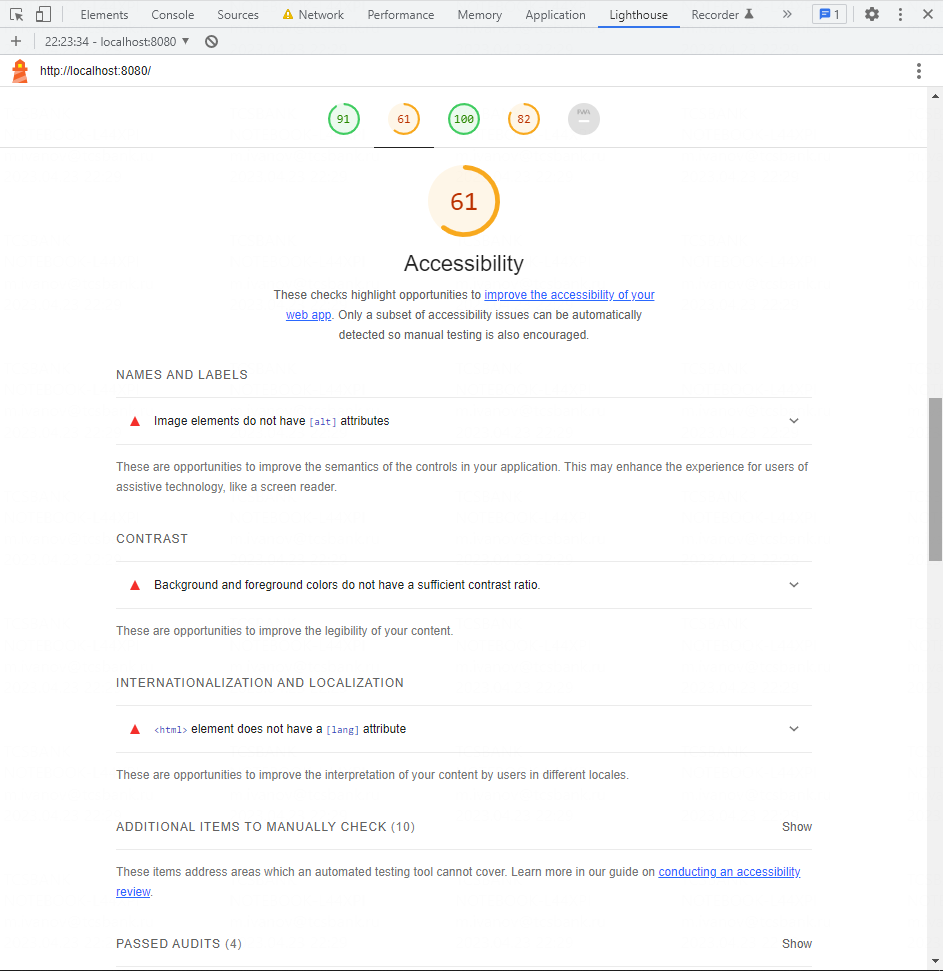


Lighthouse

Предоставляет данные об оценке производительности, доступности, соблюдения лучших практик, а так же SEO (доступности для поисковых роботов)



Каждый блок помимо оценки содержит рекомендации по улучшению метрики



Ресурсы

<https://developer.mozilla.org/en-US/docs/Web/Performance/Critical_rendering_path>

<https://web.dev/fast/>

<https://web.dev/vitals/>

<https://web.dev/user-centric-performance-metrics/>

<https://web.dev/lcp/>

<https://web.dev/optimize-lcp/>

<https://web.dev/cls/>

<https://web.dev/optimize-fid/>

<https://web.dev/fid/>

<https://web.dev/optimize-fid/>

<https://developer.chrome.com/docs/devtools/>

<https://developer.chrome.com/docs/lighthouse/overview/>

<https://www.youtube.com/watch?v=GNAENzKdciQ&list=PLAwxTw4SYaPmKmNX-INgcxQWf30KuWa_A>

<https://www.youtube.com/watch?v=AQqFZ5t8uNc>

<https://www.youtube.com/watch?v=fWoI9DXmpdk>

<https://www.youtube.com/watch?v=WxYpdw5ELrU>

<https://www.youtube.com/watch?v=t8YBZLjL-KU>