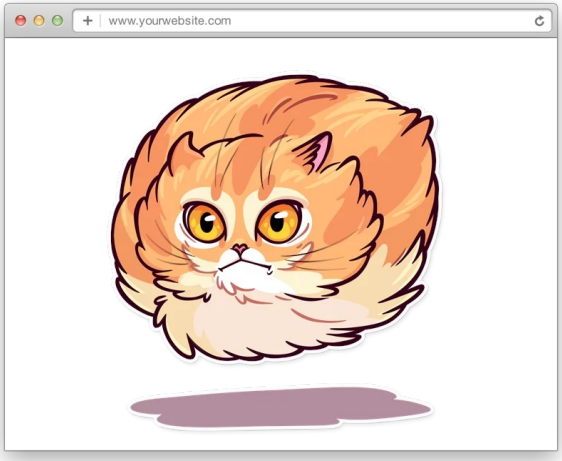
Critical Rendering Path and Performance metrics

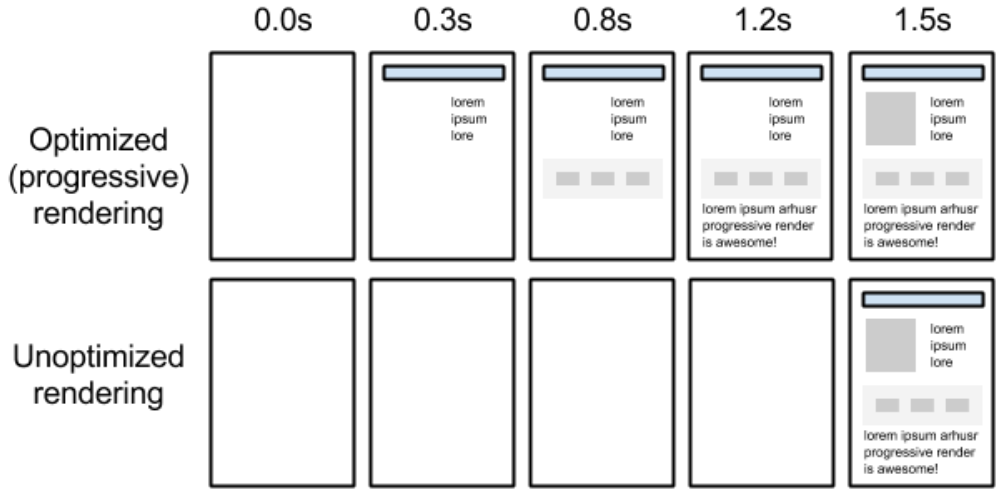
Pavel Uvarov
Senior Developer

Why

- Высокопроизводительные сайты вызывают больше доверия у пользователей и повышают удобство их использования.
- Показатели Web Vitals так же влияют на ранжирование сайтов в поисковой выдаче.
- При построении современных приложений важно уметь измерить, оптимизировать и отслеживать скорость работы ваших приложений.

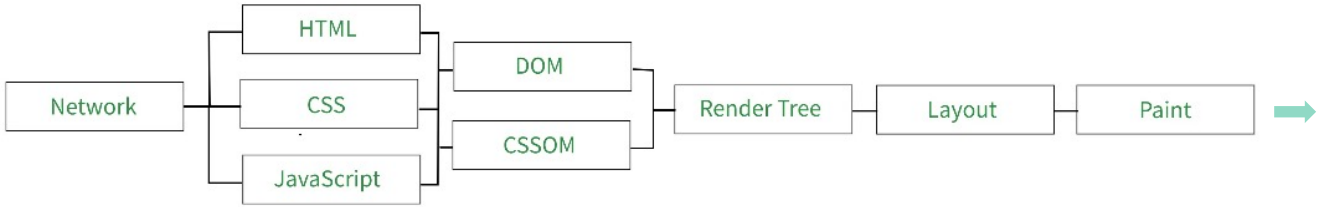


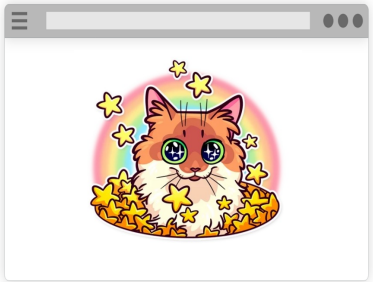
- При визуализации страниц браузер проделывает огромную работу, которую мы, веб-разработчики, обычно не замечаем.
- Понимание основ CRP важно при разработке высокопроизводительных интерактивных приложений



CRP steps

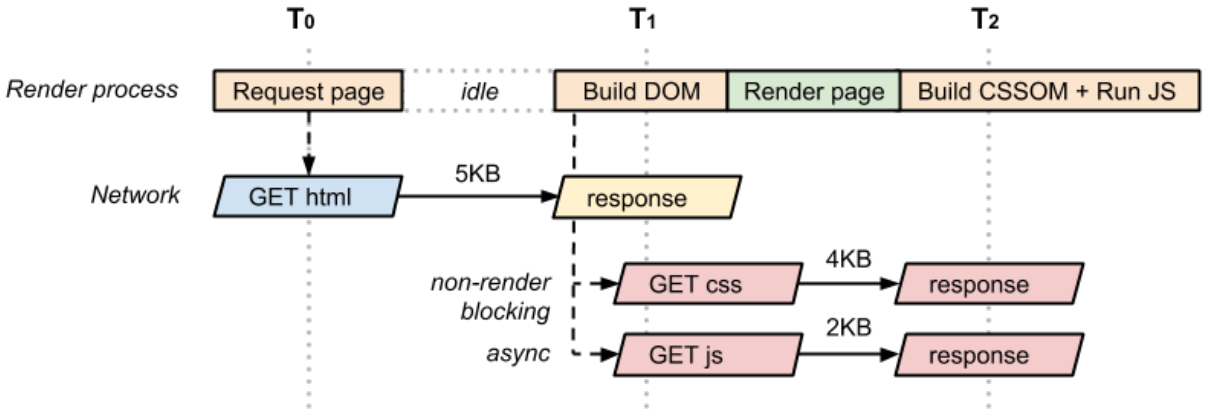
CRP - это набор шагов, которые браузеру необходимо выполнить чтобы конвертировать полученные HTML, CSS и JS в видимый для нас «живой» сайт.





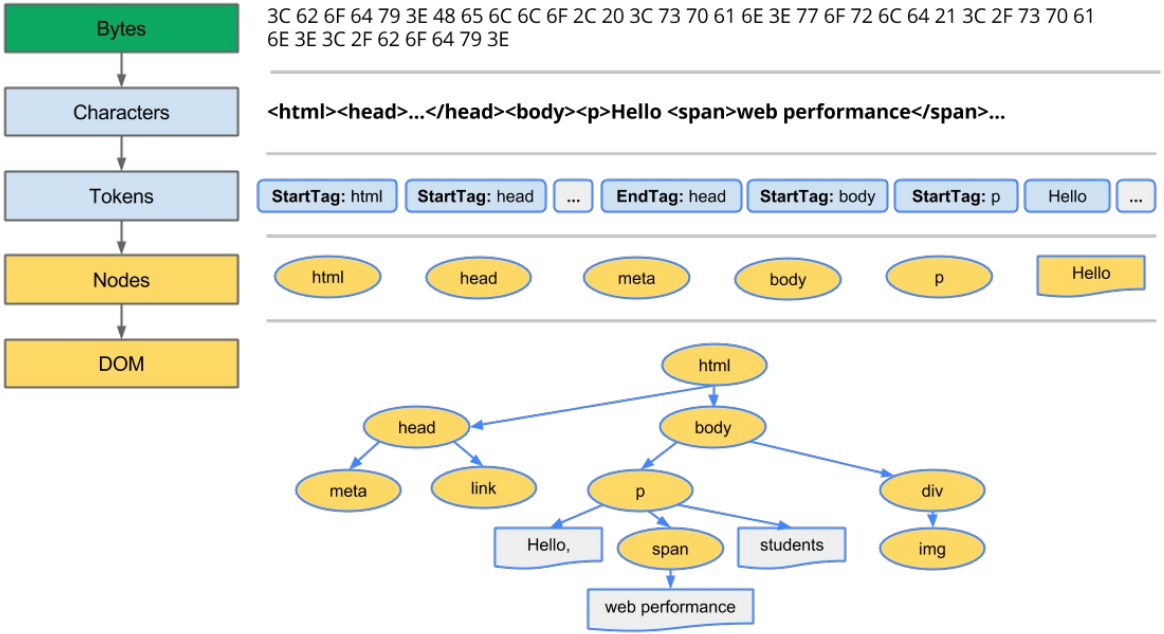
CRP steps

- Загрузка веб-страницы или приложения начинается с запроса HTML.
- Браузер начинает парсить загружаемый HTML
- Находя ссылки на внешние ресурсы, браузер создаёт новый запрос
- Синхронный JavaScript останавливает разбор DOM

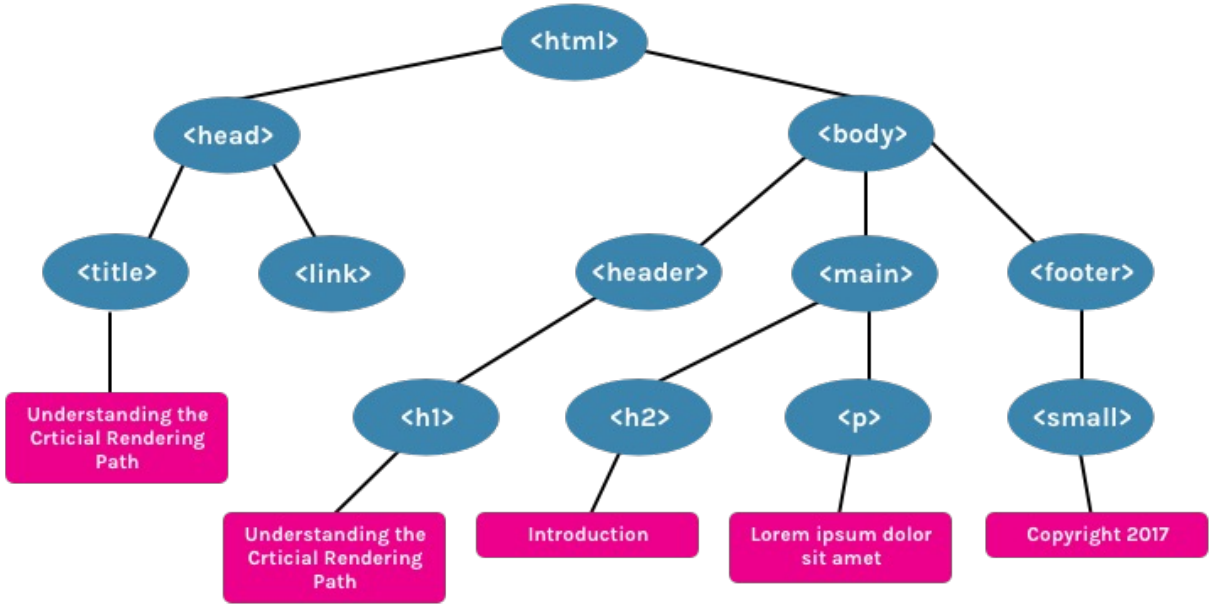


Document Object Model

- Построение DOM происходит инкрементально.
- Ответ в виде HTML превращается в токены, затем в узлы (nodes), которые формируют DOM дерево.
- Чем больше количество узлов имеет приложение, тем дольше происходит формирование DOM tree

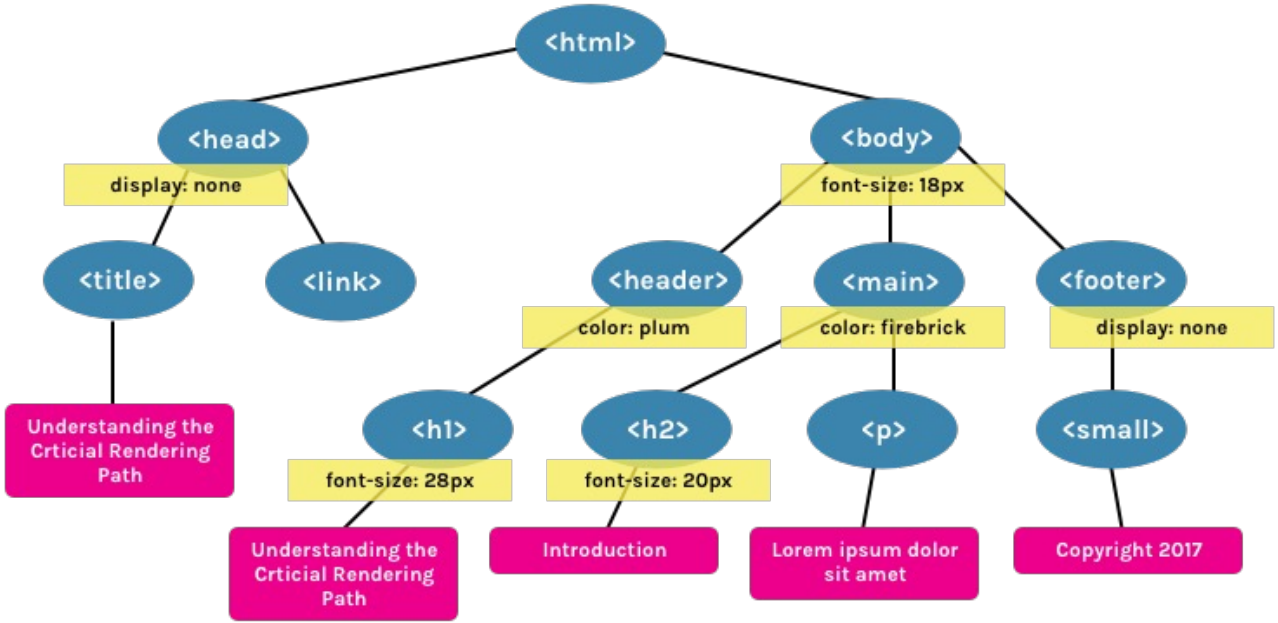


На данном шаге мы сформировали DOM дерево:

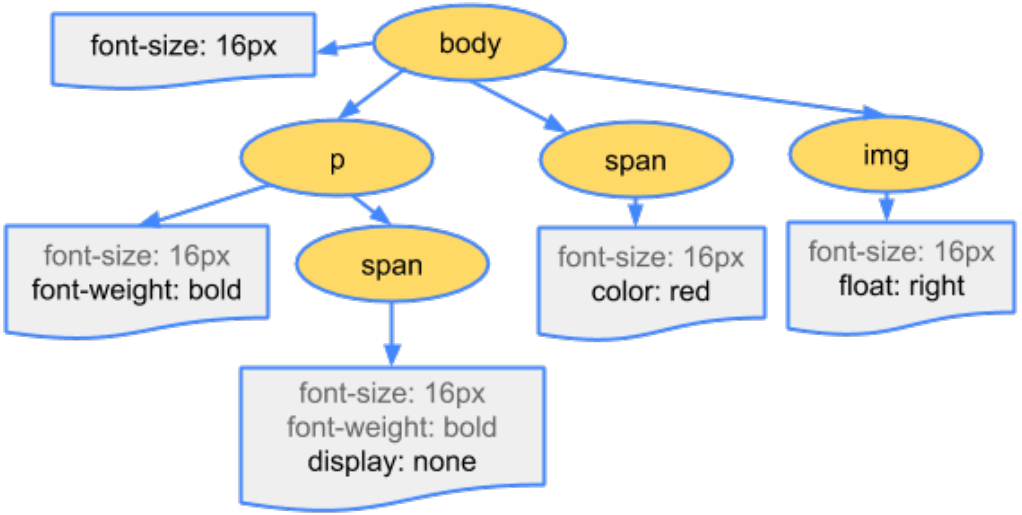


CSS Object Model

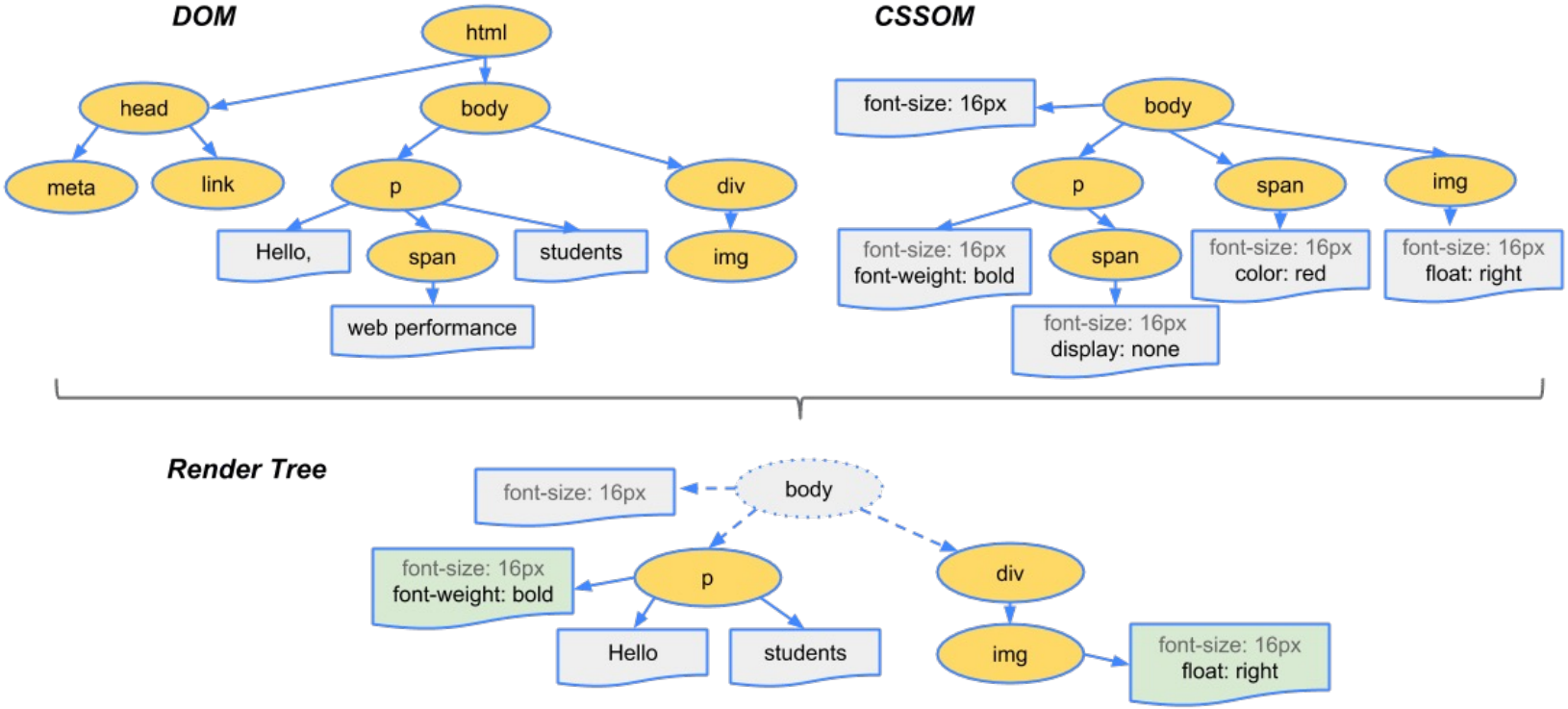
После завершения парсинга DOM, браузер конструирует CSS модель.



- Если формирование DOM инкрементально, CSSOM - нет.
- Браузер блокирует рендеринг страницы до тех пор, пока не получит и не обработает все CSS-правила.

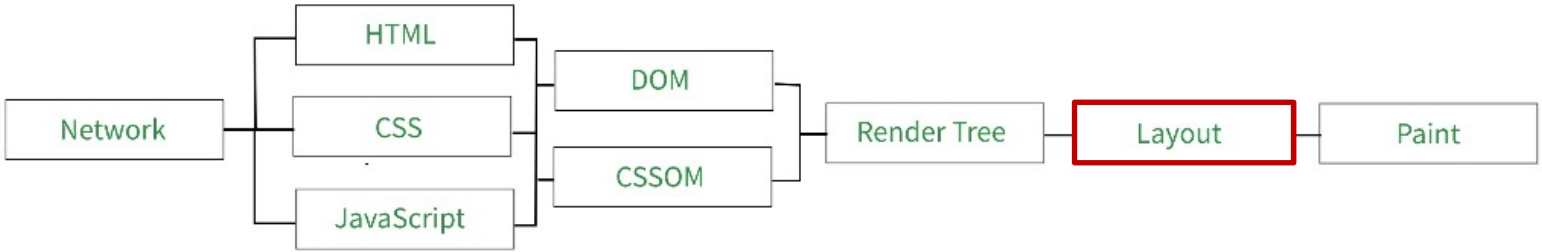


DOM и CSSOM деревья комбинируются в одно дерево

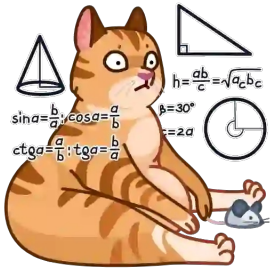


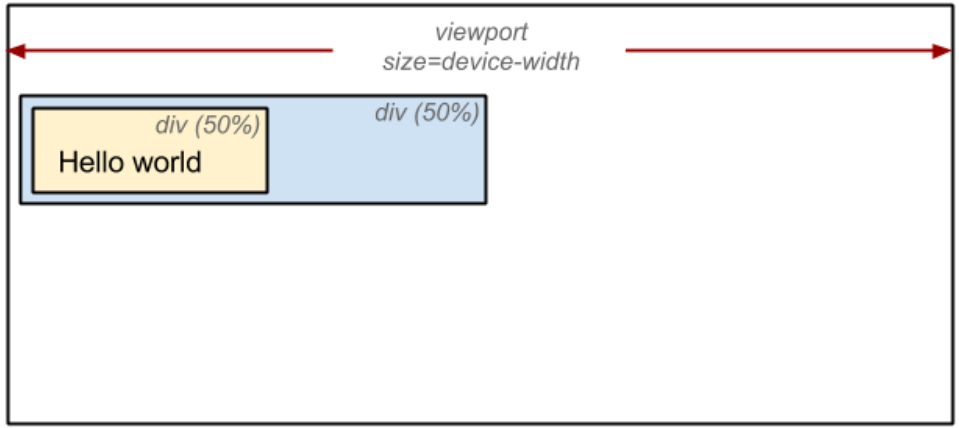
Layout

- Браузер объединяет DOM и CSSOM, формируя модель визуализации.
- Модель визуализации (render tree) содержит только те объекты, которые нужны для вывода страницы на экран.
- Теперь можно приступить к формированию макета.

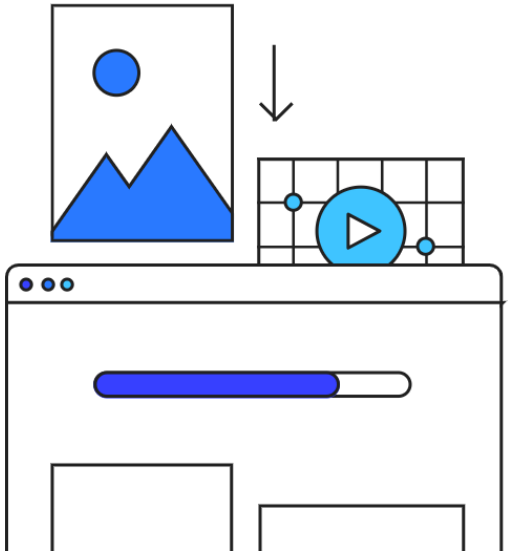


- Компоновка зависит от размеров экрана.
- Этот этап определяет, где и как на странице будут спозиционированы элементы и каковы связи между элементами.
- Каждый раз, когда дерево рендера (render tree) модифицируется запускается компоновка.

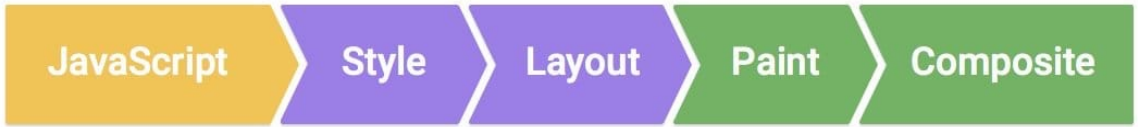




- Когда дерево рендера (render tree) создано, компоновка (layout) произошла, могут быть отрисованы пиксели.
- При первичной загрузке документа (onload) весь экран будет отрисован.
- После этого будут перерисовываться только необходимые к обновлению части эк

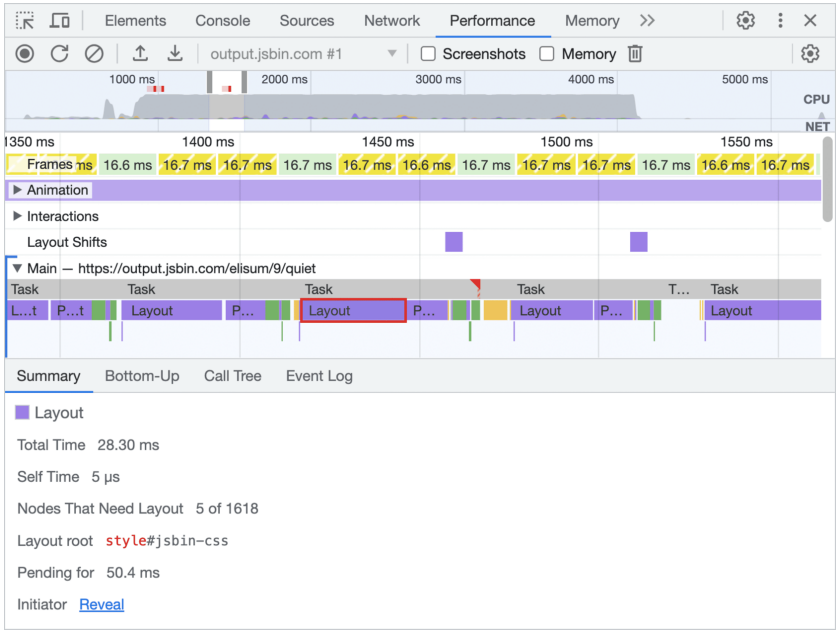


Запускается JavaScript, затем происходит перерасчёт стилей, layout и композиция



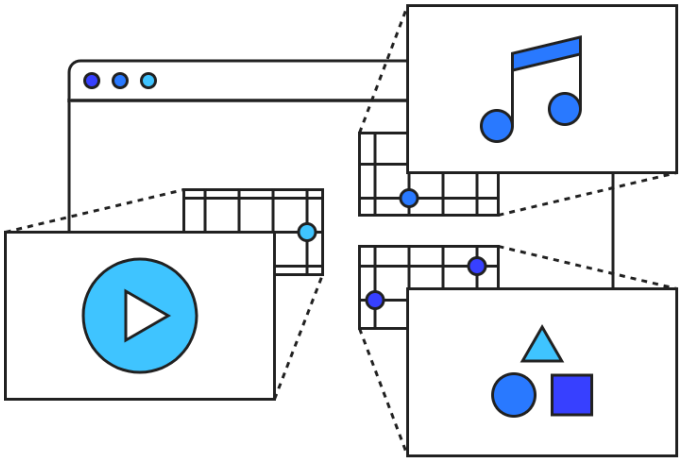
Reflow

- Reflow означает перерасчёт позиций и геометрии элементов .
- Изменения «геометрических значений», таких как width, height, left, или top требуют перерасчёт layout
- Скорость reflow зависит от размера DOM



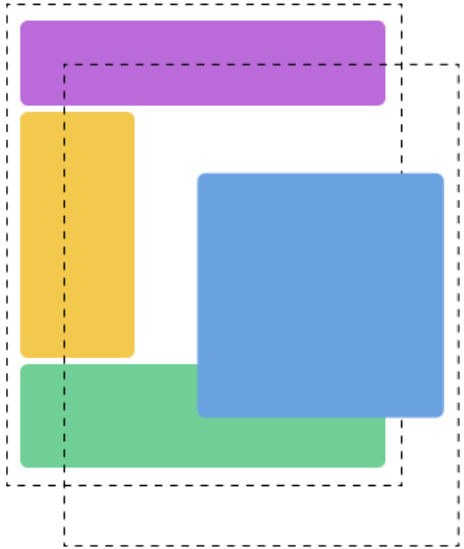
Repaint

Вызвать Repaint могут изменения свойств color, background, visibility, которые не изменяют размеров и положения элемента



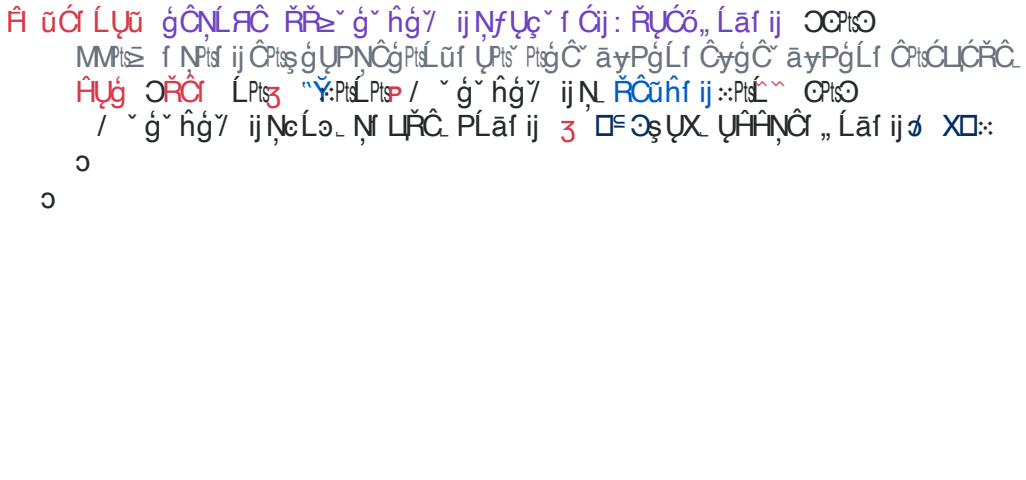
Composite

На этапе Composite группирует различные элементы по слоям, отрисовывает пиксели и затем объединяет эти слои в готовую страницу в отдельном потоке композитора



Layout thrashing

- **Layout thrashing – форсированная синхронизация layout – когда перерасчёт макета происходит синхронно**
- **Это происходит когда вы обновляете стили в JS и затем сразу же их считываете**

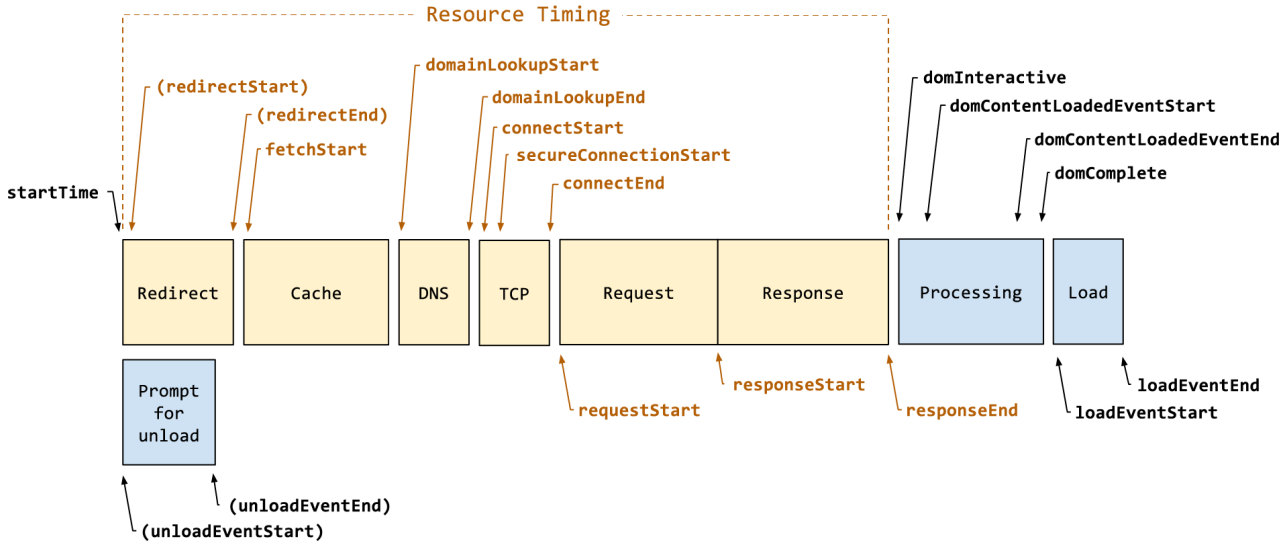


Общие рекомендации по оптимизации CRP

- Уменьшайте количество критических ресурсов
- Оптимизируйте количество необходимых запросов, а так же размеры файлов.
- Критические ресурсы должны загружаться в первую очередь
- Избегайте больших женности элементов

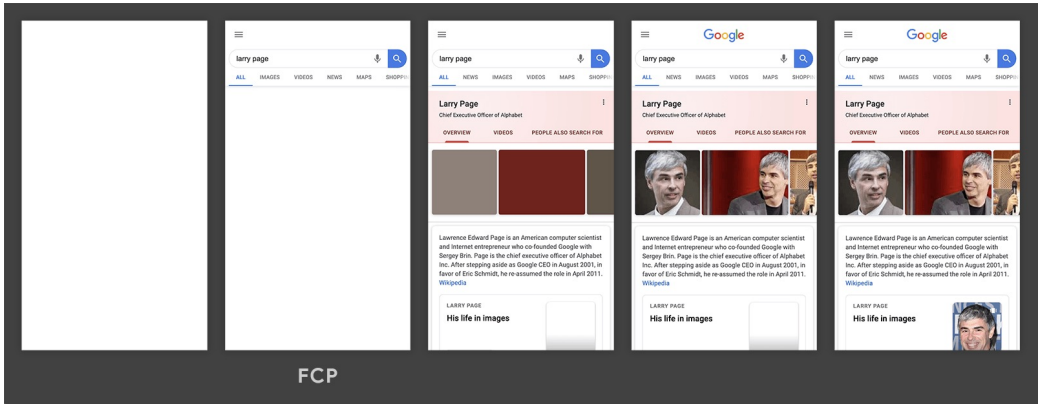
- Time to First Byte (TTFB)
- First Contentful Paint (FCP)
- Largest Contentful Paint (LCP)
- First Input Delay (FID)
- Time to Interactive (TTI)
- Cumulative Layout Shift (CLS)
- Interaction to Next Paint (INP)

TTFB – это метрика, которая определяет время между запросом на ресурс и первым байтом респонса



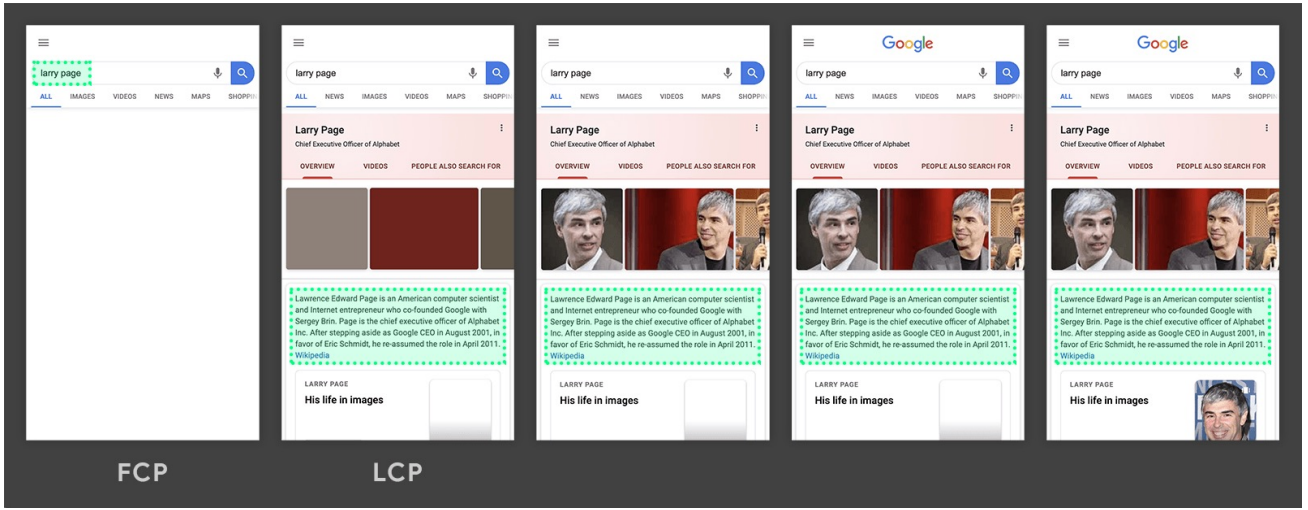
What is FCP?

- The First Contentful Paint (FCP) – определяет скорость появления первого контента на странице с момента загрузки.
- Под «контентом» понимается текстовые блоки и изображения (а так же svg и не non-white canvas)



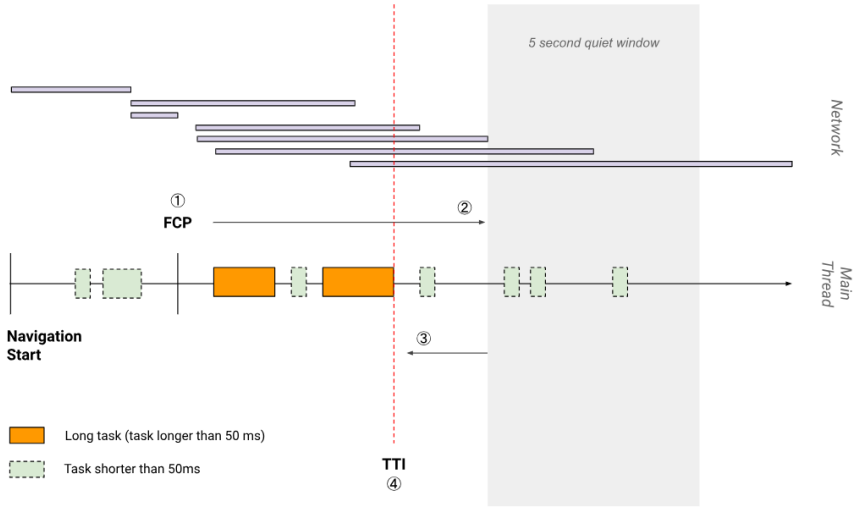
Largest Contentful Paint (LCP)

The Largest Contentful Paint (LCP) представляет собой время, которое требуется для загрузки самого большого элемента на странице, такого как изображение, текст или другой видимый контент.



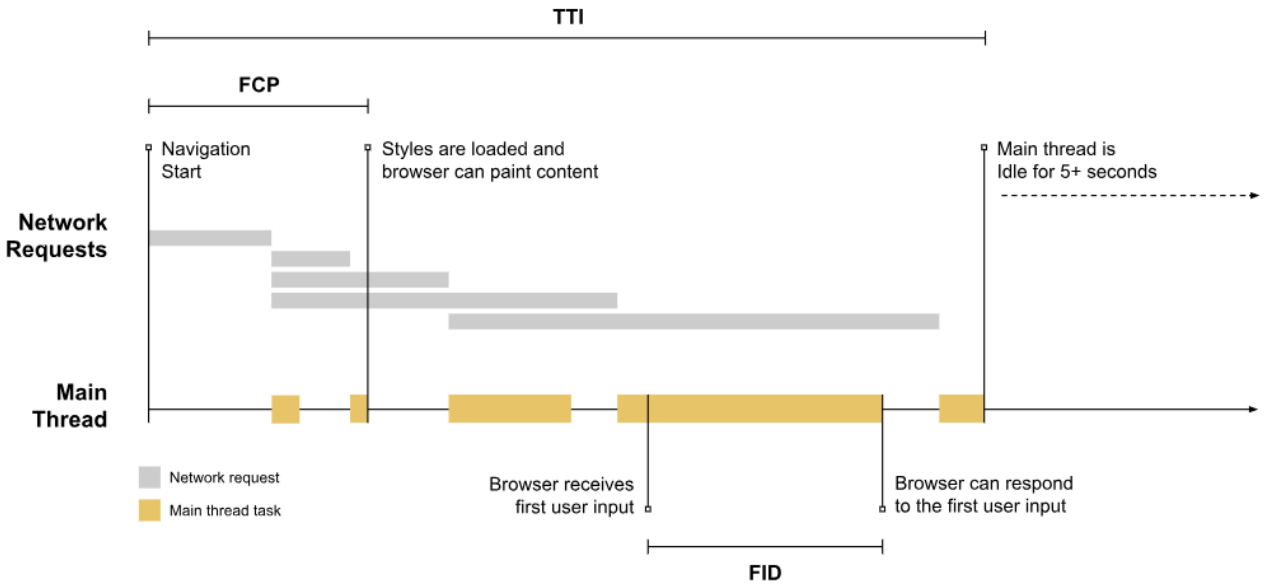
Time to Interactive (TTI)

- TTI определяет сколько времени нужно на то, чтобы страница стала полностью интерактивной
- TTI – это время окончания последней тяжёлой таски, после которой происходит «затишье»



First Input Delay (FID)

- First Input Delay определяет отзывчивость и скорость работы пользовательского интерфейса
- FID обычно возникает между FCP и TTI



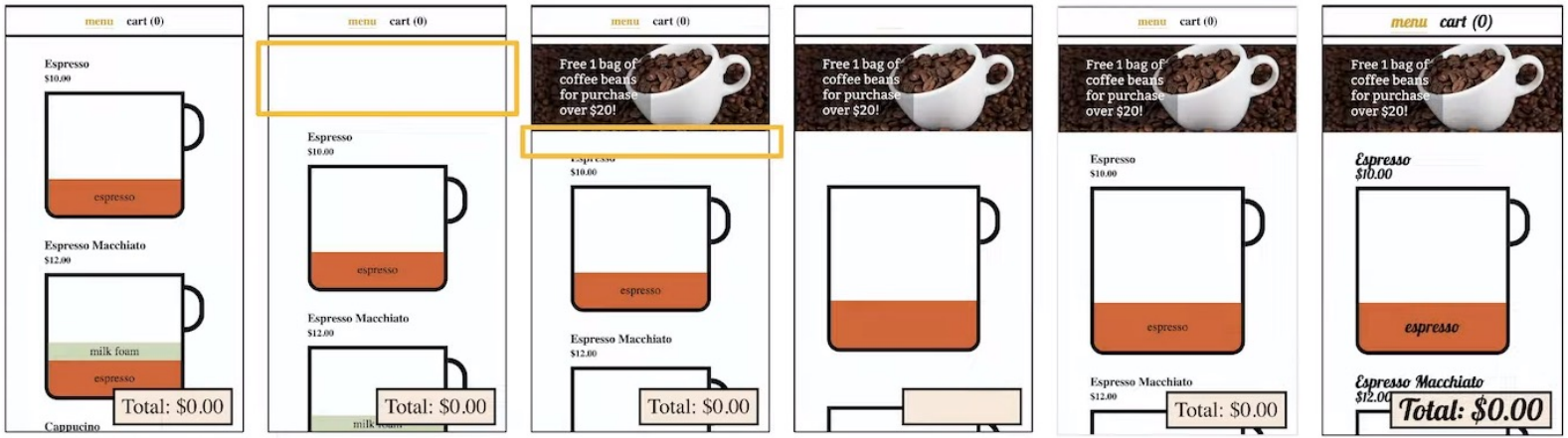
Interaction to Next Paint (INP) – определяет промежуток времени между тем, когда пользователь инициирует взаимодействие и моментом рисования следующего кадра

Poor responsiveness

Good responsiveness

Cumulative Layout Shift (CLS)

- Измеряет смещение всех элементов, которое происходит независимо от действий пользователя.
- CLS рассчитывается как сумма всех вертикальных сдвигов контента на странице за время загрузки



Core Web Vitals

(Loading)

LCP

Largest Contentful Paint

(Interactivity)

FID

First Input Delay

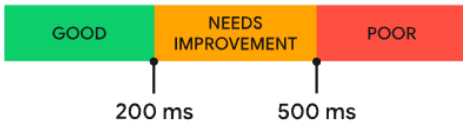
(Visual Stability)

CLS

Cumulative Layout Shift

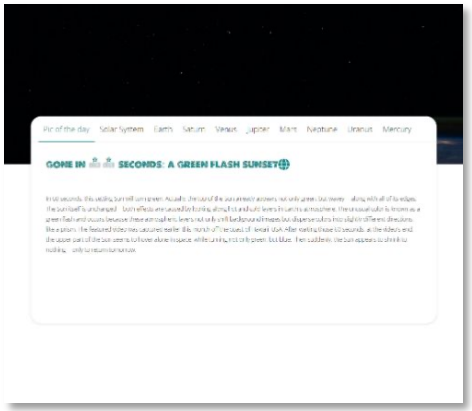
INP

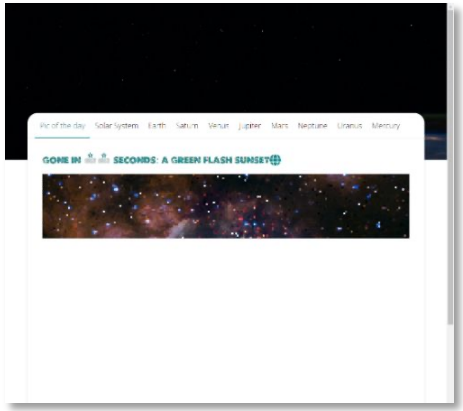
Interaction to Next Paint

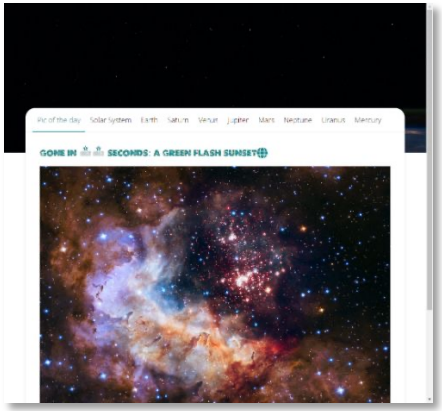


Demo before

Rendering before







First Contentful Paint 1.2 s	Largest Contentful Paint 7.0 s
Total Blocking Time 40 ms	Cumulative Layout Shift 0.42

- Обеспечьте быструю доставку ресурсов
- Откладывайте загрузку non-critical CSS
- Оптимизация изображений и шрифтов (размеры, формат)
- Используйте lazy-load и preload
- Используйте Server-side rendering (SSR)
- Оптимизируйте third-party ресурсы

example

(Visual Stability)

CLS

Cumulative Layout Shift

- Устанавливайте явные размеры для контента
- Избегайте CSS анимации которые вызывают reflow

example

(Interactivity)

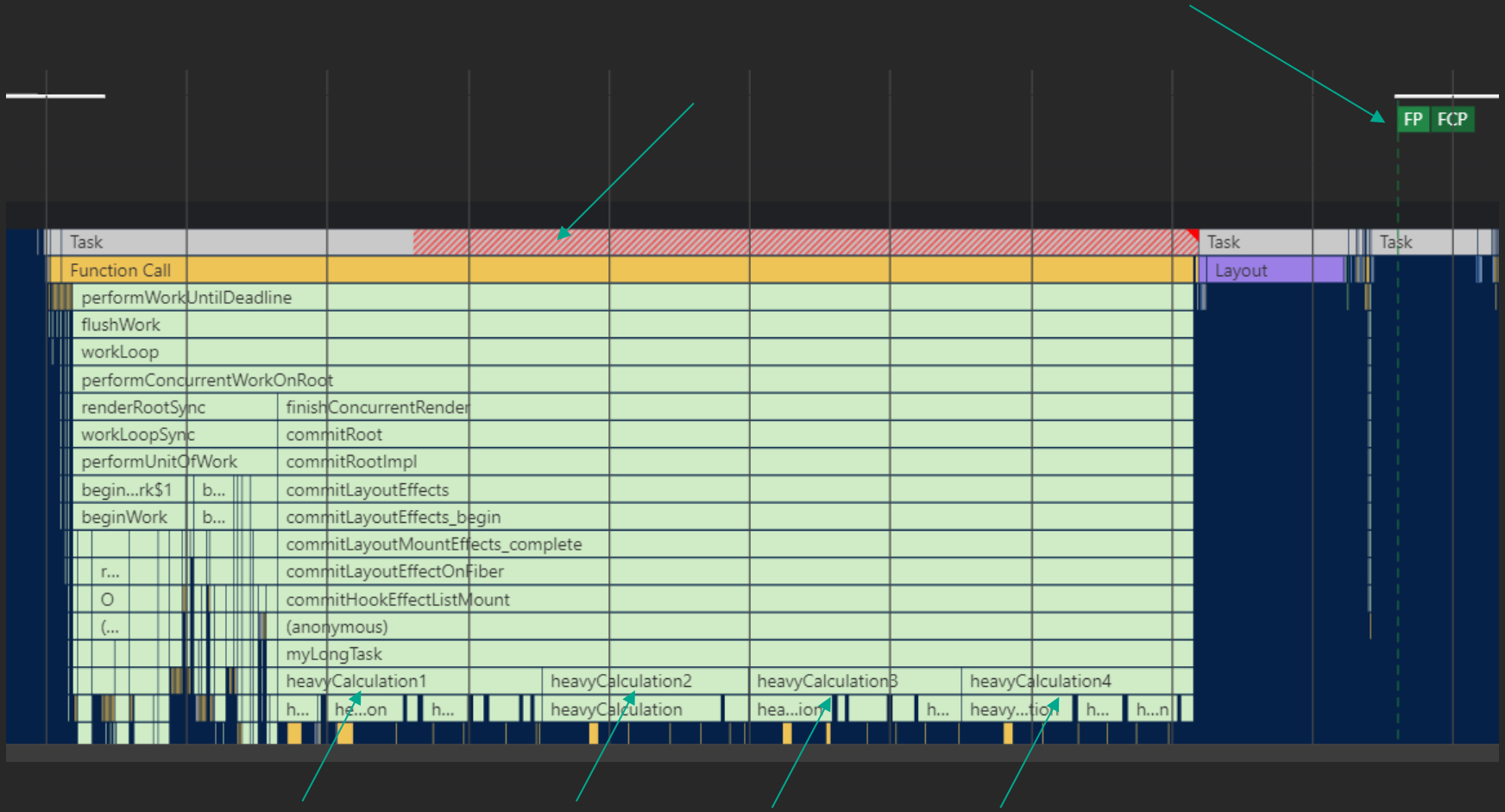
FID

First Input Delay

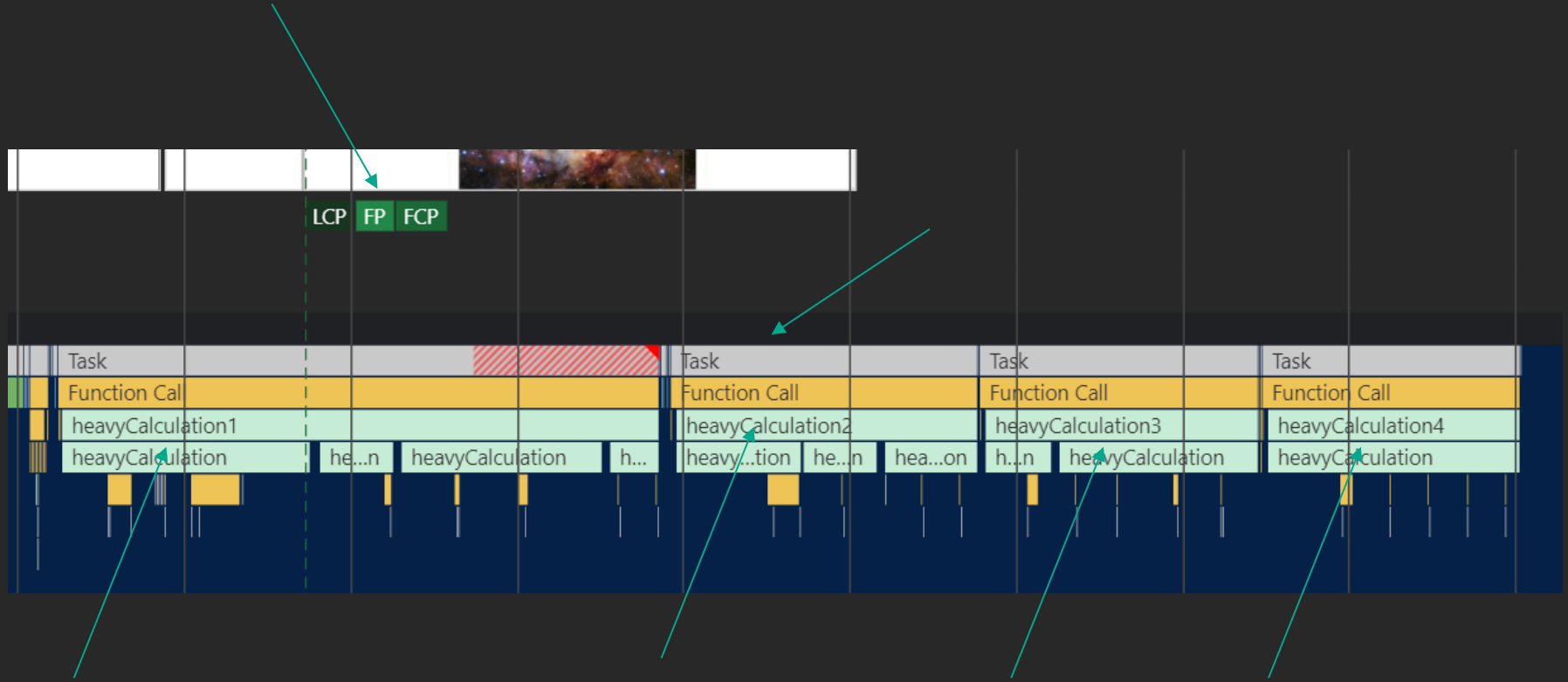
- Разбивайте тяжёлые задачи на части
- Используйте современный JS, для современных браузеров
- Разбивайте JS на чанки через code splitting
- Удалите неиспользуемый код
- Избегайте больших перерисовок

example

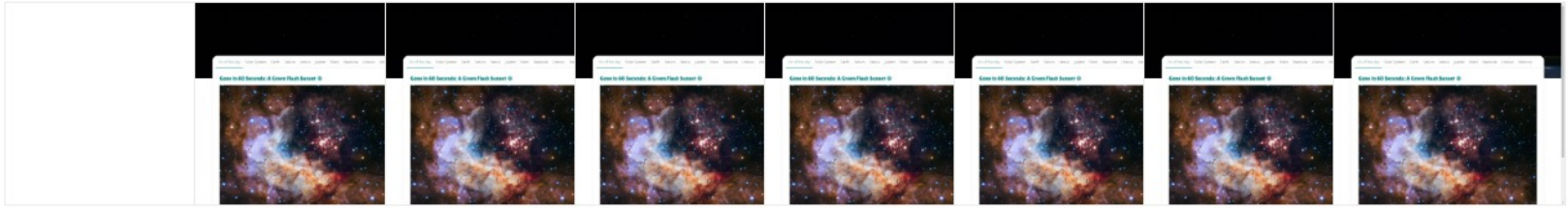
Long task before



Long task after



Before/After



Before/After

52

Performance

- First Contentful Paint: 1.2 s
- Total Blocking Time: 40 ms
- Largest Contentful Paint: 7.0 s
- Cumulative Layout Shift: 0.42

95

Performance

- First Contentful Paint: 0.4 s
- Total Blocking Time: 30 ms
- Largest Contentful Paint: 1.4 s
- Cumulative Layout Shift: 0.006

Resources

CRP
https://web.dev/critical-rendering-path-analyzing-crp/
https://developer.mozilla.org/ru/docs/Web/Performance/Critical_rendering_path
https://blog.logrocket.com/jank-free-page-loading-with-media-aspect-ratios/
https://habr.com/ru/articles/445264/

Repaint/Reflow
High-performance animations.
https://gist.github.com/paulirish/5d52fb081b3570c81e3a
https://dev.to/gopal1996/understanding-reflow-and-repaint-in-the-browser-1jbg
https://docs.google.com/spreadsheets/d/1Hvi0nu2wG3oQ51XRHtMv-A_ZlidnwUYwgQsPQUg1R2s/pub?single=true&gid=0&output=html

Optimize Web Vitals:
https://web.dev/articles/font-best-practices
https://developer.chrome.com/blog/performance-insights/
https://web.dev/articles/serve-modern-code-to-modern-browsers
https://web.dev/articles/user-centric-performance-metrics
https://web.dev/articles/efficiently-load-third-party-javascript
https://web.dev/articles/optimize-long-tasks

Спасибо за внимание!

Pavel Uvarov

@spiderpoul

kaspersky