Web Performance Optimisation

(706.041 Information Architecture and Web Usability 3VU WS 2021/2022)

Florian Marcher, Paul Höfler, Vera Tysheva, Group 3

24 November 2021

Overview

- General
- Load Time
- Run Time
- Quick Wins

Motivation

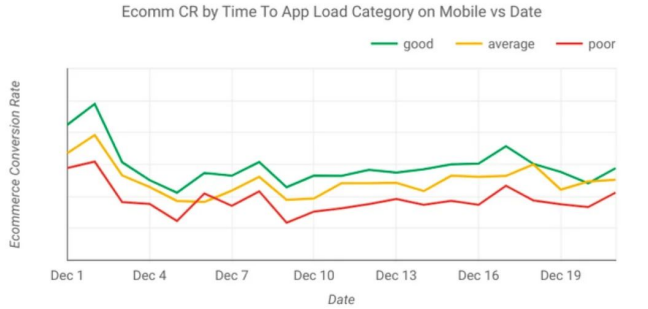
- Retaining users
- Improve conversions
- Improve user experience
- Influences page ranking

“Pinterest reduced perceived wait times by 40% and this increased search engine traffic and sign-ups by 15%.”

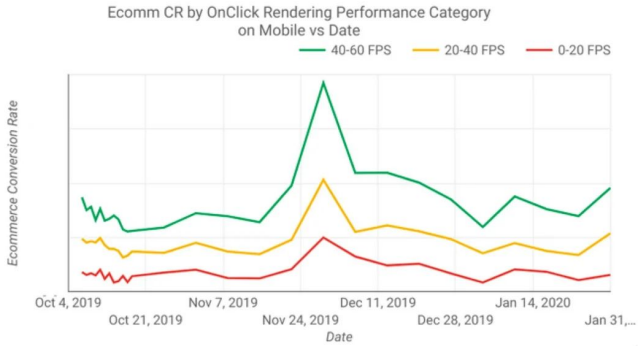
“When AutoAnything reduced page load time by half, they saw a boost of 12% to 13% in sales”

web.dev/why-speed-matters/
medium.com/@Pinterest_Engineering/driving-user-growth-with-performance-improvements-cfc50dafadd7
digitalcommerce360.com/2010/08/19/web-accelerator-revs-conversion-and-sales-autoanything/

Motivation



Conversion rate by time to application load (react.hydrate)



Conversion rate by rendering performance

smashingmagazine.com/2021/01/front-end-performance-2021-free-pdf-checklist/

Response Times

- 0 to 16 ms: smooth response (10ms for app 6ms for browser draw).
- < 100 ms: immediate response, feeling of action → reaction.
- 100 ms to 1 s: feel of progress, longer tasks like loading page.
- ≥ 1 s: users lose focus.
- ≥ 10 s: users are frustrated, might not come back.

Users react to delay with annoyance

web.dev/rail/

Planning and Metrics

- Run performance tests regularly.
- Goal: Be at least 20% faster than your fastest competitor.
- Gather performance data:
  - Synthetic
  - Real user monitoring
- Choose build tools and framework:
  - Lightweight vs bloated
- Client vs Server side rendering.

smashingmagazine.com/2021/01/front-end-performance-2021-free-pdf-checklist/

Important Milestones

- First Contentful Paint (FCP)
  - Loading start until first rendered parts of content
- Time to Interactive (TTI)
  - Loading start until website becomes interactive (reacts to input)
- Cumulative Layout Shift (CLS)
  - Movement of objects after initial display
    - Can be misused in dark patterns

Are all important in Google's ranking algorithm

web.dev/cls/ web.dev/fcp/ web.dev/tti/
ingosteinke.medium.com/optimizing-speed-and-usability-for-googles-core-web-vitals-9db93606d335

Performance Budget

- Predefined set of limits on metrics that affect site performance.

- Quantity-based:
  - Size of files, number of resources.
- Timing-based:
  - First contentful paint, time to interactive.
- Rule-based:
  - Performance scores like WebPage or Lighthouse.

Ideally, use all of them.

web.dev/performance-budgets-101/

smashingmagazine.com/2021/01/front-end-performance-2021-free-pdf-checklist/

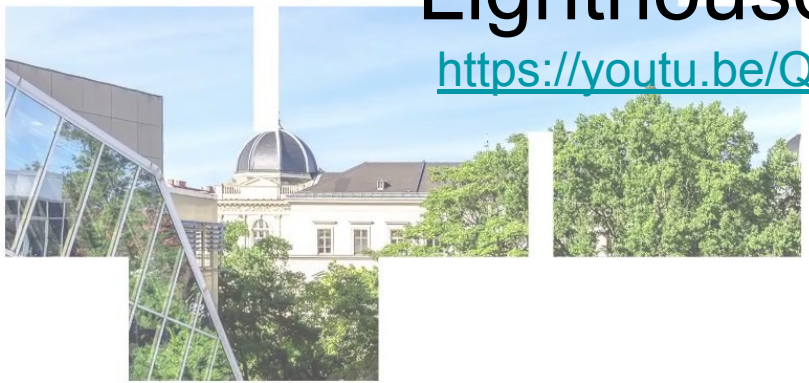
Performance Tools

- Chrome Dev Tools
  - Simulates throttling
  - View paint events in real-time
  - Monitor resource usage
- Lighthouse
  - Included in Chrome
  - Simulated mid range device with slow connection
  - Reports load and response times
  - Gives suggestions
- WebPageTest
  - Can use different devices

web.dev/rail/

Lighthouse demo

https://youtu.be/QkHjxY77-bs



Load Time

Use Minification and Compression

- Turn on minification.
- Turn on Compression.
- Disable for:
  - Compressed Images/PDFs: doesn’t improve size by much if already compressed
  - Files <= 1500 bytes: transmitted in a 1500 byte packet anyways (size of MTU)
- Gzip most widespread
  - Brotli (by Google) is getting more common

blog.hubspot.com/website/gzip-compression

computerworld.com/article/2693941/why-it-doesn-t-make-sense-to-gzip-all-content-from-your-web-server.html

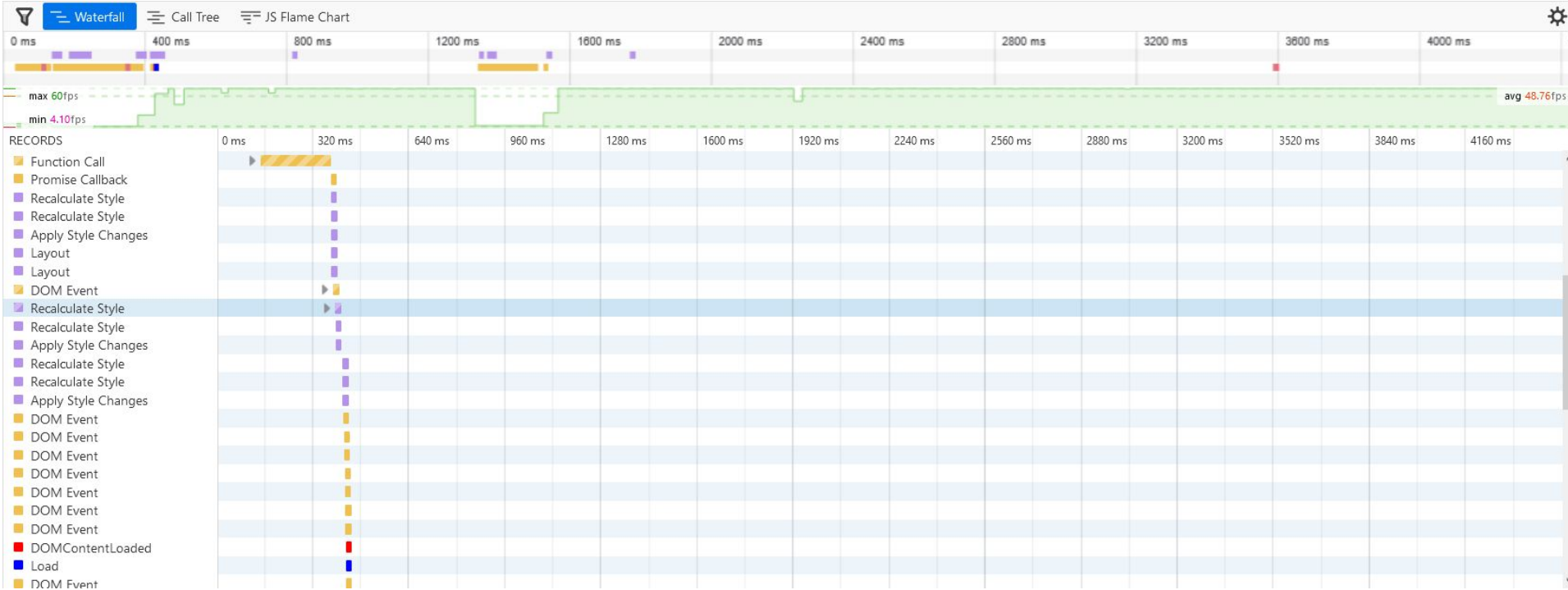
Optimize JavaScript Loading

- Browsers block rendering and downloading on JS download and execution.
- No parallel JS file download by default.
  - To preserve execution order.
- Use tree shaking to eliminate dead code.
  - Technique to only import what’s needed and eliminate from code on build.

Souders, Steve. Even faster web sites: performance best practices for web developers. " O'Reilly Media, Inc.", 2009.

developers.google.com/web/fundamentals/performance/optimizing-javascript/tree-shaking#what_is_tree_shakingv

Optimize JavaScript Loading



PRPL Pattern

- **Push** critical resources using preload
- **Render** initial page
- **Pre-cache** non-critical resources
- **Lazy-load** remaining resources on demand

web.dev/rail/

Souders, Steve. Even faster web sites: performance best practices for web developers. " O'Reilly Media, Inc.", 2009.

Async vs Defer

- Async
  - Loads and executes scripts in the background.
  - Ignores script order
  - Non-blocking
  - For independent scripts
- Defer
  - Moves scripts to the bottom of the page.
  - Retains script order
  - Executes after DOM is ready but before DOMContentLoaded event.

- Page should be usable without scripts

javascript.info/script-async-defer

HTTP 2/3

- Make sure it is turned on
- Both vastly improve performance
- HTTP3 Improves performance for slow networks.
- HTTP3 QUIC instead of TCP
  - UDP-based
  - Suffers less from packet loss.
  - Multiple streams
  - Connection migration

smashingmagazine.com/2021/08/http3-core-concepts-part1/

rfc-editor.org/rfc/rfc9000.html

Caching Strategy

- Use a far-future EXPIRES header (where applicable).
  - Images and Scripts don’t change often.
- Service worker interface comes with a Cache interface.
  - Cache is only cleared if browser exceeds it’s storage limit. Updates are your responsibility.

developer.mozilla.org/en-US/docs/Web/HTTP/Caching

developers.google.com/web/ilt/pwa/caching-files-with-service-worker

Optimizing Images

- Images are half of the size of typical websites.
- Lossy
  - JPEG
  - Reduces image quality
- Lossless
  - PNG, GIF
  - Remove metadata
  - Use Tools
- Use new formats like WebP, AVIF or JPEG XL.
  - Performance vs support

developer.mozilla.org/en-US/docs/Web/Media/Formats/Image_types

Static Files and Prerendering

- Prefer static files, they are fastest to serve.
  - Can be generated at build → Static Rendering
- Prerendering is capturing an apps initial state in static HTML and using JS to fill in the gaps.
- The more is static (not generated on server or browser) the faster the page appears.

developers.google.com/web/updates/2019/02/rendering-on-the-web

Run Time

JavaScript Performance

- Focus on “expensive” parts → use profiling
- Remove unused JavaScript
- Avoid memory leaks → use profiling
- Save repeatedly used DOM elements in variables

nodesource.com/blog/improve-javascript-performance/

Selective use of Animation

- Page feels faster if done correctly.
  - Slower if incorrectly
- Via SVG, video, JavaScript or CSS
  - Limited by file size or CPU performance.

developer.mozilla.org/en-US/docs/Web/Performance/Animation_performance_and_frame_rate

RAIL

- User-centric performance model.
- Response - Animation - Idle - Load
- Goals: Key performance metrics related to UX; persistent since based on human perception.
- Guidelines: Recommendations to achieve goals, might be specific to hardware, therefore change over time.

web.dev/rail/

Quick Wins

- Audit first.
- Use static files or prerender.
- Check cache, compression and resource hints are setup properly.
- When using pre-built JS frameworks, choose wisely.
- Optimize images.
- Trim, optimize, minify, defer and lazy-load assets.

smashingmagazine.com/2021/01/front-end-performance-2021-free-pdf-checklist/#quick-wins

Thank you