



Fast Websites

Евгений Лейченко, 21 мая 2013



Задача

1. Пользователь должен тратить меньше времени на ожидание, чем сейчас

А зачем это нужно?

1. Чем быстрее пользователь начнёт работу - тем лучше
2. Каждая секунда ожидания уменьшает потенциальные продажи [\[*\]](#)
3. Google, скорость загрузки страниц и SEO [\[*\]](#)
4. Каждый запрос - DNS lookup + HTTP request + headers + 4J [\[*\]](#)

http://www.gomez.com/pdfs/wp_why_web_performance_matters.pdf

А чем мы можем помочь?



80-90% of the end-user response time is spent on the frontend. Start there.

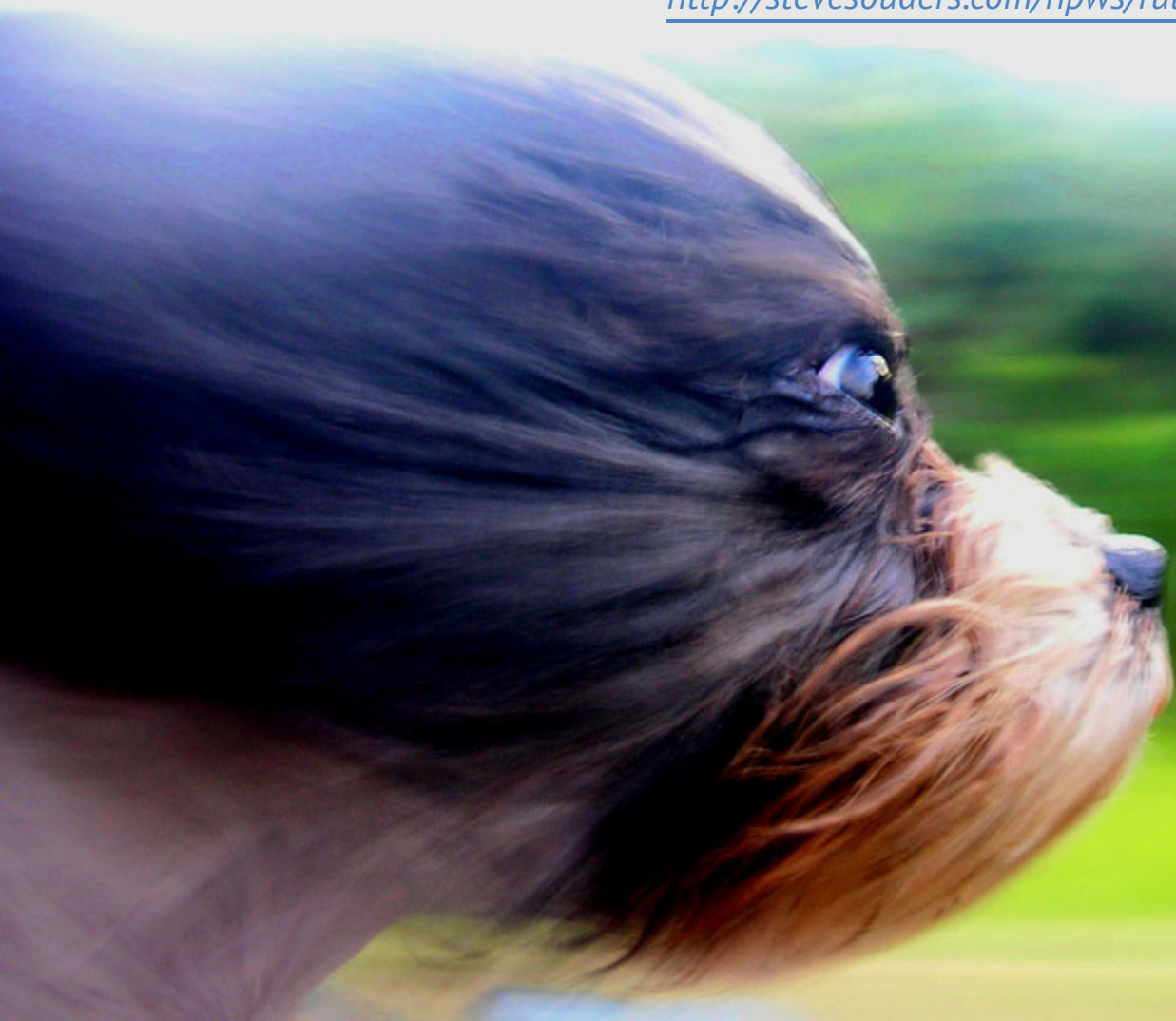
Steve Souders, <http://www.stevesouders.com/blog/2012/02/10/the-performance-golden-rule/>

А что почитать?

- [Steve Souders "High Performance Web Sites"](#)
- [Steve Souders "Even Faster Web Sites"](#)
- [Stoyan Stefanov "Web Performance Daybook Volume 2"](#)
- [Peter Smith "Professional Website Performance"](#)
- [Nicholas Zakas "High Performance JavaScript"](#)

14 правил от Steve Souders

<http://stevesouders.com/hpws/rules.php>



14 правил от Steve Souders...

1. Make Fewer HTTP Requests
2. Use a Content Delivery Network
3. Add an Expires Header
4. Gzip Components
5. Put Stylesheets at the Top
6. Put Scripts at the Bottom
7. Avoid CSS Expressions

...14 правил от Steve Souders

1. Make JavaScript and CSS External
2. Reduce DNS Lookups
3. Minify JavaScript
4. Avoid Redirects
5. Remove Duplicated Scripts
6. Configure ETags
7. Make AJAX Cacheable

Сжимаем JavaScript

- [Closure Compiler Service](#) (advanced mode)
- [YUI Compressor](#)
- [UglifyJS](#) (javascript)
- [/packer/](#)
- [r.js](#) (AMD)

Сжимаем CSS

- [YUI Compressor](#)
- [r.js](#) (AMD)
- [CSS Compressor](#) (online)
- [Minify CSS](#) (online)
- [Robson CSS Compressor](#) (online)
- [Sqwish](#) (node.js)

Сжимаем графику

- [pngcrush](#), [pngout](#), [OptiPNG](#), [Image Worsener](#)
- [jpegtran](#), [jpegoptim](#)
- [gifsicle](#)
- [Smush.it](#) (online)

Сжимаем HTML

- [HtmlCompressor](#)
- [WebMarkupMin](#) (.NET)
- [HTML Minifier](#)

Сжимаем шрифты

- [Font Squirrel](#)

Сжали. Что дальше?

- Lazy loading, contextual loading [\[*\]](#)
- Deferred parsing
- COMET, Web Sockets
- Web Workers
- CSS images and fonts
- Fast JavaScript & CSS [\[*\]](#)
- ...

Инструменты

A top-down view of a wooden workbench covered with a variety of hand tools. The tools are arranged somewhat haphazardly, showing signs of use and age. On the left, there's a large set of sockets and a few open-end wrenches. In the center, several combination wrenches are scattered, along with a few long, thin rods or bolts. To the right, a large hammer with a wooden handle, a hand saw, and a few more wrenches are visible. A small, green-handled tool, possibly a screwdriver or a small wrench, is also present. The wooden surface is light-colored and shows some wear and tear. The overall scene suggests a workshop or a place where manual labor is performed.

PageSpeed

The screenshot shows the Google PageSpeed Insights tool interface. At the top, there's a navigation bar with tabs: Elements, Resources, Network, Sources, Timeline, Profiles, Audits, Console, and PageSpeed. Below this are 'Refresh' and 'Clear' buttons. The main content area is divided into a left sidebar and a main panel.

Overview

The page [Белорусский портал TUT.BY](#) got an overall PageSpeed Score of **76** (out of 100). [Learn more](#)

Suggestion Summary

Click on the rule names to see suggestions for improvement.

- High priority (3)**
 - Enable compression
 - Serve resources from a consistent URL
 - Enable Keep-Alive
- Medium priority (6)**
 - Leverage browser caching
 - Minify JavaScript
 - Combine images into CSS sprites
 - Optimize images
 - Defer parsing of JavaScript
 - Serve scaled images
- Low priority (9)**
 - Inline Small CSS
 - Optimize the order of styles and scripts
 - Specify a cache validator
 - Minify HTML
 - Minify CSS
 - Specify image dimensions
 - Put CSS in the document head
 - Remove query strings from static resources
 - Specify a Vary: Accept-Encoding header

High priority. These suggestions represent the largest potential performance wins for the least development effort. You should address these items first:
[Enable compression](#), [Serve resources from a consistent URL](#), [Enable Keep-Alive](#)

Medium priority. These suggestions may represent smaller wins or much more work to implement. You should address these items next:
[Leverage browser caching](#), [Minify JavaScript](#), [Combine images into CSS sprites](#), [Optimize images](#), [Defer parsing of JavaScript](#), [Serve scaled images](#)

Low priority. These suggestions represent the smallest wins. You should only be concerned with these items after you've handled the higher-priority ones:
[Inline Small CSS](#), [Optimize the order of styles and scripts](#), [Specify a cache validator](#), [Minify HTML](#), [Minify CSS](#), [Specify image dimensions](#), [Put CSS in the document head](#), [Remove query strings from static resources](#), [Specify a Vary: Accept-Encoding header](#)

Already done! There are no suggestions for these rules, since this page already follows these best practices. Good job!

YSlow

[Home](#)
[Grade](#)
[Components](#)
[Statistics](#)

Rulesets **YSlow(V2)** [Edit](#) | [Help](#)

Grade D Overall performance score 66 Ruleset applied: YSlow(V2) URL: <http://www.tut.by/>

ALL (23) FILTER BY: [CONTENT \(6\)](#) | [COOKIE \(2\)](#) | [CSS \(6\)](#) | [IMAGES \(2\)](#) | [JAVASCRIPT \(4\)](#) | [SERVER \(6\)](#)

[Tweet](#) [Share](#)

F	Make fewer HTTP requests
F	Use a Content Delivery Network (CDN)
A	Avoid empty src or href
F	Add Expires headers
F	Compress components with gzip
B	Put CSS at top
A	Put JavaScript at bottom
E	Avoid CSS expressions
n/a	Make JavaScript and CSS external
D	Reduce DNS lookups
A	Minify JavaScript and CSS
A	Avoid URL redirects
A	Remove duplicate JavaScript and CSS
F	Configure entity tags (ETags)
A	Make AJAX cacheable

Grade F on Make fewer HTTP requests

This page has 13 external Javascript scripts. Try combining them into one.
This page has 3 external stylesheets. Try combining them into one.
This page has 14 external background images. Try combining them with CSS sprites.

Decreasing the number of components on a page reduces the number of HTTP requests required to render the page, resulting in faster page loads. Some ways to reduce the number of components include: combine files, combine multiple scripts into one script, combine multiple CSS files into one style sheet, and use CSS Sprites and image maps.

[»Read More](#)

Copyright © 2013 Yahoo! Inc. All rights reserved.

Online-инструменты

- [Pingdom](#)
- [GTmetrix](#)
- [WebPagetest](#)
- Google Analytics

HTTP Archive

- Текущая статистика
- История
- Тренды

Спасибо!

- email: e.leychenok@gmail.com
- twitter: [yleichanok](https://twitter.com/yleichanok)
- skype: [e.leychenok](https://www.skype.com/people/e.leychenok)