Performance Contest - ZEROfy.de

I'm submitting my (very) small personal site which has launched 2 years ago. I know, there are only 6 pages, but that's something I'm kind of proud of;)

URL: https://zerofy.de/

The before-situation

Just before you published the contest I was just done with my last optimization, which drove me crazy for the last weeks. So there are unfortunately no before-screenshots or page speed scores.

A long time ago my aim was the 100/100 Google score. It doesn’t mean a thing, but it looks just great and satisfying. Furthermore, when a potential customer checks out your site, he’ll get a hint of my skills.
Technical set up

Server
I’m hosting my site at netcup and don’t even have a dedicated server, only lame shared-ip hosting. I got a very good price for 15 domains + 10GB of space (it’s like $15 a year!!!) so I never switched, which would boost my performance even more.

Netcup shared hosting is pretty shitty: You can’t enable any keep-alive connections, there are like 50 other domains in your ip neighborhood and the worst part: No HTTP/2 support.

CMS
My initial plan was a smashingmagazine/moz-style knowhow section, where users would get state of the art articles about SVG, Pagespeed and SEO stuff. So the blog engine Wordpress would be pretty nice, right? But after starting the first article I realised the HUGE amount of work which has to be done to finish a (very good) article. So that’s WIP.

Wordpress is a bit of a noob-proof CMS. It includes all kind of jquery, emoji, shortlink and feed link stuff into your wp_head.

So I’ve installed Contact Form 7, the most used form plugin for wordpress. And I also tried a social icons plugin and the analytics plugin.

Elements
When the site was ready for launch, it contained:

- Video: A .mp4 and a .webm hero video
- JS: Full screen overlay loader, the twenty/twenty beforeafter script, a polygon styled and mouse-move-sensitive navbar, social media JS, contact form JS, Wordpress JS (emoji, embed), bootstrap JS, own JS file
- Images: 2 before/after images (big resolution), a few (huge, detailed) SVGs, fallback JPGs
- Fonts: I’ve hosted them locally (Libre Baskerville and Source Sans Pro) to avoid the initial Flash of Unstyled Text, FontAwesome (embedded using the <link> tag in the <head>
The before-results
As I can remember there were like 90-100 requests with a loading time (pingdom test tool) of 2-3s and a page size of 5-6 MB. At this time, I was pretty satisfied with the pagespeed. When the loader faded out, the whole site was fully loaded and it was beautiful.

The shock came when a friend of mine entered my URL on his phone (3G network) and my stupid **loader did not vanish**. The window.load (or .ready) function prevented the loader from fading out because some scripts/resources were still loading, even when the site was ready to use.

At this moment I’ve started to realize the importance of site performance and the mighty render blocking magic.

The now-situation
The server, the CMS, the images, the video, everything is the same. But I’ve tweaked every component to get the most out of it.

Server

Hosting
No changes at all. The hoster refused to enable HTTP/2 or keep-alive and I haven’t switched to another hoster (soon).

Caching, Minify, GZIP, jQuery
The usual suspects got the support from the W3 Total Cache plugin, which does a great job, even in the free version. The most difficult part was to establish a good order of minified JS files in order to ensure jQuery and the rest will work as intended.

jQuery: This library is pretty big, so I’d love to get rid of it in the first place. Unfortunately the before/after script needs jQuery. At least I defered the library, so it doesn’t block the initial rendering.
**Analytics.js:** The only pain in the ass is Google’s Analytics itself. This known caching “problem” could be resolved using a script which updates a local version of analytics. Because it doesn’t affect the pagespeed at all, I haven’t done this yet.

**Wordpress stuff**

I’ve unregistered every action I won’t need including the whole wp_head cleanup, I’ve dismissed the social plugin and the Contact Form 7 plugin. Instead I’m using simple, branded share buttons:

![Share Buttons](image)

**Contact Form**

The contact form got replaced by an own written php form using the PHP mail function and some AJAX, so I don’t need another confirmation subpage. This gave me savings in the amount of ~10-15 requests and way more flexibility.

![Contact Form](image)
Media / Requests

Images/Video

The video had a length of 25s and a file size of 3-5 MB. So I’ve cropped it to 8 seconds and looped it using an invisible fade. The new sizes are 1MB (OGV) and 2.3MB (MP4). The OGV format is the most effective video source. It starts loading the first 2 seconds, then playback it and after the site is already complete it loads the rest of the OGV, check the waterfall:

So my fallback order of video <source> files is: .OGV > .WEBM > .MP4.

Mobile fallback: There is also a function which prevents the video to load on mobile devices to safe bandwidth (if not mobile -> append data-src to <source> and load video).

Fonts

To avoid render blocking fonts, I’ve switched from the <link> implementation in the <head> to the recommended webfontloader. So the fonts are now loading afterwards. Of course, I could switch to Arial, Helvetica system fonts (9 requests less), but my branding (business cards for example) is set to Libre Baskerville and Source Sans Pro.

FontAwesome has been replaced with a customized icomoon set of ~10 icons to save at least 130 KB. Comparison:

<table>
<thead>
<tr>
<th>Sizes</th>
<th>TTF</th>
<th>CSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FontAwesome</td>
<td>162 KB</td>
<td>31 KB</td>
</tr>
<tr>
<td>Customized Icomoon</td>
<td>2 KB</td>
<td>1 KB</td>
</tr>
</tbody>
</table>
Images

HTTP/2 problem: Because my hoster does not allow HTTP/2, I have a huge disadvantage regarding pagespeed. Using HTTPS (Google ranking factor) reduces the speed performance. So first I had to outsource my images and videos to cloudinary. Cloudinary uses the HTTP/2 protocol allowing multiple file downloads at the same time.

The .webp format: I’ve switched from PNG to JPG first. Then I’ve added the WEBP format to my new <picture> element. The new picture tag is very mighty but needs a lot of work as you can see in the following picture:

```html
\c<picture>
\source media="(min-width:992px)" src="https://res.cloudinary.com/dd5pj2ex/raw/upload/v15076537993/shop-optimierung-vorher_2x_aa2pp6.jpg">
\source media="(min-width:992px)" src="https://res.cloudinary.com/dd5pj2ex/raw/upload/v15076536393/shop-optimierung-vorher_2x_chlvuy_webp" type="image/webp">
\source media="(min-width:360px)" src="https://res.cloudinary.com/dd5pj2ex/raw/upload/v15001806713/shop-optimierung-vorher_2x_768x566_bjbbx1_webp" type="image/webp">
\source media="https://res.cloudinary.com/dd5pj2ex/raw/upload/v1500178016/shop-optimierung-vorher_2x_m942nc_webp 1.5x" type="image/webp">
\source media="https://zeroxy.de/wp-content/uploads/2015/12/shop-optimierung-vorher.jpg,
https://zeroxy.de/wp-content/uploads/2015/12/shop-optimierung-vorher@2x.jpg 1.5x">
\img width="1053" height="625" src="https://zeroxy.de/wp-content/uploads/2015/12/shop-optimierung-vorher.jpg alt="shop optimierung vorher" class="twentytwenty-before" style="clip: rect(0px 526.5px 626px 0px);">
\</picture>
```

There are media queries for different sizes, DPI based images and a .JPG fallback.

SVG: From the beginning, SVGs were my first choice. I’m sick of stock footage and SVGs allow me to have a single file for every possible resolution, and it’s sharp as hell. The major problem: I was using SVG animations and the browser support for embedded animations is pretty thin. The solution: Inline-SVGs using SMIL (Synchronized Multimedia Integration Language). A nice side benefit of using inline SVGs is, that it reduces the amount of requests (because it’s right in the DOM).

Rendering

Critical CSS

To boost the speed index score and to get the fastest experience as possible, I’ve created a separate .PHP file which contains all css styles affecting the above the fold view. This file gets included in the head to ensure it gets loaded right with the DOM.
Barba.js

Barba.js is a tiny script which allows you to create page transitions without reloading the whole site. When switching a page, the pre defined wrapper gets replaced by the wrapper of the requested page. Push state is used then for changing the URL in the user’s browser. Barba.js boosts the user’s *perceived* page speed. I had to slow down the page using a SVG page transition (the prefetch function caused a too fast loading). Thanks to the transition, I also have time to scroll back to top and re-init the scripts (like the polygon-navigation, the before/after slider and the contact form).

The after-results

Webpagetest.org mobile test

With all the tweaks I get the following results (tested with webpagetest.org, 3G Fast, Frankfurt, Moto 4G with Chrome. Median of 5 runs):

<table>
<thead>
<tr>
<th>Requests</th>
<th>28 (First view 17)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page Size</td>
<td>569 KB</td>
</tr>
<tr>
<td>TTFB</td>
<td>720 ms</td>
</tr>
<tr>
<td>Start render time</td>
<td>1.066 s</td>
</tr>
<tr>
<td>Speed Index</td>
<td>1521</td>
</tr>
</tbody>
</table>

Results link

Webpagetest.org desktop test

Results tested with webpagetest.org, Cable 5/1, Frankfurt, Chrome. Median of 5 runs:

<table>
<thead>
<tr>
<th>Requests</th>
<th>31 (First view 17)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page Size</td>
<td>1,612 KB</td>
</tr>
<tr>
<td>TTFB</td>
<td>197 ms</td>
</tr>
<tr>
<td>Start render time</td>
<td>0.300 s</td>
</tr>
<tr>
<td>Speed Index</td>
<td>870</td>
</tr>
</tbody>
</table>

Results link
Desktop - Pingdom
Results tested with tools.pingdom.com, Stockholm (testid: bOwnRF)

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load time</td>
<td>362 ms</td>
</tr>
<tr>
<td>Page Size</td>
<td>1.7 MB</td>
</tr>
<tr>
<td>TTFB</td>
<td>127 ms</td>
</tr>
<tr>
<td>Grade</td>
<td>99</td>
</tr>
<tr>
<td>Requests</td>
<td>28</td>
</tr>
</tbody>
</table>

Lighthouse (Chrome audits)
I haven’t tested the PWA section because nobody needs a PWA for this kind of page.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance Score</td>
<td>85</td>
</tr>
<tr>
<td>First meaningful paint</td>
<td>1.410 ms</td>
</tr>
<tr>
<td>Perceptual Speed Index</td>
<td>2.522</td>
</tr>
<tr>
<td>Estimated Input Latency</td>
<td>50 ms</td>
</tr>
</tbody>
</table>

Report link