GREEN CODING Guide by VAST FORWARD

Green Coding in web development is a fundamental part of Vast Froward's sustainability strategy. Not only are we reducing the energy demand and greenhouse gas emissions within our company this way, but also across our entire supply chain.

Extrapolated to servers and devices worldwide, each line of code has the potential to lower the energy demand and emissions. If developers make their code just a bit more energy-efficient, the scalability effects can achieve a tremendous amount.

1. Media (images and videos)

- ☐ Do not embed images larger than necessary for display.
- ☐ Responsiveness through the use of the 'Picture Element' or 'srcset'
- ☐ Adjusting image qualities. If the system allows, let the adjustment be automatic. [1]
- ☐ Use current image formats like 'WEBP'. [2]
- ☐ Load images only when needed (Lazy Load), and limit the number of images overall.
- ☐ Combine icons into a sprite to minimize server requests.
- ☐ Use a 'CDN' for a large number of users. [3]
- ☐ Use videos in 'MP4' or 'WebM' with current codecs while paying attention to small file sizes.
- ☐ Avoid autoplay in videos

Note: The CMS Wordpress automatically generates different sizes of embedded images, which are often not needed or used – these can be disabled.
Through plugins like Better-Image-Sizes, required image sizes can be generated on the fly and only in the needed sizes.

[1] Choose between 60-70% or an optimal balance between image size and quality.
[2] CMS, e.g. Contao, have the capability to convert images to WEBP and deliver them. For Wordpress, various plugins can be used for this purpose, e.g. Images to WEBP as a free option (https://wordpress.org/plugins/images-to-webp/).
[3] Content Delivery Networks (CDNs) can manage the flow of content and reduce the physical distance between the server and the user.

2. Scripts

- ☐ Merge and minimize JavaScript (goal ≈ 100 kB). To achieve this, it can also be split up and only included where it is needed – code splitting. [4]
- ☐ Remove unused JavaScript, e.g., via tree shaking. [5]
- ☐ Check if there are smaller libraries for the use case that simplify the complexity of the code or enable the desired function in another way.
- ☐ Avoid having multiple instances of modules in JavaScript.
- ☐ Switch the system if too many scripts are necessary for simple functions.

3. CSS

- ☐ Merge and minimize CSS files into a single file.
- ☐ Load CSS late and insert critical CSS into the head section. [6]
- ☐ Bootstrap and similar libraries are helpful for fast development but are rarely fully utilized. Cut unnecessary definitions or define styles yourself. E.g. CSS Grid is now supported by all common browsers.

4. Web Fonts

- ☐ Ideally, use system fonts.
- ☐ Reduce the number of web fonts and check which ones are actually necessary.
- ☐ Icon fonts are often unnecessarily large. Reduce to only needed icons and deliver these as SVG or Sprite (also reduces loaded CSS, since in addition to the font, extra CSS is loaded).
- ☐ Webfonts can be reduced to the necessary characters through *font subsetting*, saving up to 80%.[7]

---

[4] In an ideal scenario, scripts are minimized in advance. If that's not possible, CMS can also take over or be enabled to do so via plugins. In Wordpress, for example, Autoptimize.
[5] https://medium.com/@netxm/what-is-tree-shaking-de7c6be5cadd
[6] Critical CSS can be generated and integrated through tools and plugins, e.g. via https://www.sitelocity.com/critical-path-css-generator.
[7] https://walterebert.com/blog/subsetting-web-fonts

5. Caching/Hosting

- ☐ Switch to a “green”, sustainable operating host.
- ☐ *Caching* can reduce server load for static files. Static files should be held for at least 1 year with appropriate caching policies.
- ☐ Most CMS can hold a static version of the page or through plugins, so servers don’t have to process them anew each time. This could also be resolved through Server-Caching. [8]
- ☐ Text compression on the server (configure the use of GZIP and ideally Brotli on the server).
- ☐ Reduce/minimize server requests

6. Other

- ☐ Reduce *DOM*-size. Onepagers often have larger data amounts and thus longer loading times. Evaluate whether dividing content into subpages is sensible.
- ☐ Do not neglect *SEO*. Content that is easier to find uses fewer resources.

7. Tools

Various tools can offer a good idea of how a website is performing and whether it is sustainable. Some also provide direct suggestions for optimization opportunities.

- Digital Beacon: Calculates the CO2 footprint of a page on first and repeated visits. However, it does not provide an overview of the entire “weight” since it does not include the complete page in the calculation.
- Ecograder: Another tool that can calculate the CO2 footprint.
- Cabin: CO2 conscious, sustainable analytics tool that also checks the CO2 footprint of each page (External tool with a subscription model).
- Google PageSpeed Insights and GTMetrix: Show the current performance of a web page and provide options for further optimisation.

[8] There are many plugins with varying amounts of functions for Wordpress. E.g. a simple and good plugin would be WP Fastest Cache.