GREEN CODING Guide by VAST FORWARD

Green Coding in der Webentwicklung ist fester Bestandteil der Nachhaltigkeitsstrategie von Vast Forward. So verringern wir den Energiebedarf und die Emission von Treibhausgasen nicht nur in unserem Unternehmen, sondern auch in unserer gesamten Lieferkette.

Hochgerechnet auf Server und Devices auf der ganzen Welt hat jede Zeile Code das Potenzial, den Energiebedarf und die Emissionen zu senken. Gestalten Entwickler ihren Code auch nur etwas energieeffizienter, ist durch Skalierungseffekte extrem viel zu erreichen.

1. Medien (Bilder und Videos)

- ☐ Bilder nicht größer einbinden, als für die Darstellung benötigt
- ☐ Responsivität durch Nutzung von Picture Element oder srcset
- ☐ Anpassung der Bildqualitäten. Wenn das System es ermöglicht, die Anpassung übernehmen lassen. [1]
- ☐ Aktuelle Bildformate wie WEBP nutzen [2]
- ☐ Bilder erst nachladen, wenn sie gebraucht werden (Lazy Load), Anzahl an Bildern allgemein begrenzen
- ☐ Icons als Sprite zusammenfassen, um Serveranfragen zu minimieren
- ☐ Bei großer Nutzerzahl ein CDN nutzen [3]
- ☐ Videos als MP4 oder WebM mit aktuellen Codecs verwenden. Auf kleine Dateigröße achten.
- ☐ Autoplay bei Videos vermeiden

Anmerkung: Das CMS Wordpress generiert von hinterlegten Bildern automatisch verschiedene Bildgrößen, die oft nicht benötigt oder verwendet werden – diese können deaktiviert werden.
Über Plugins wie Better-Image-Sizes können benötigte Bildgrößen on the Fly und auch nur in den benötigten Größen generiert werden.

---

[1] 60-70% bzw. ein optimales Maß zwischen Bildgröße und Bildqualität wählen
[2] CMS wie z.B. Contao besitzen die Möglichkeit, Bilder in WEBP zu konvertieren und auszuspielen. Für Wordpress können dafür diverse Plugins verwendet werden, z.B. Images to WEBP (https://wordpress.org/plugins/images-to-webp/) als eine kostenlose Variante
[3] Content Delivery Networks (CDNs) können den Ablauf von Inhalten verwalten und die physische Entfernung zwischen Server und Nutzer verringern.

2. Skripte

- [ ] Javascript zusammenführen und minimieren (Ziel ≈ 100 kB). Um das zu erreichen, kann es auch aufgeteilt werden und nur da eingebunden werden, wo es benötigt wird – Code Splitting) [4]
- [ ] ungenutztes Javascript entfernen z.B. über Tree Shaking [5]
- [ ] Prüfen, ob es für den Anwendungsfall kleinere Bibliotheken gibt, die die Code-Komplexität vereinfachen oder eine gewünschte Funktion auf andere Weise ermöglichen
- [ ] mehrfach vorhandene Module im Javascript vermeiden
- [ ] Falls für einfache Funktionen zu viele Skripte notwendig sind: System wechseln

3. CSS

- [ ] CSS Dateien in einer einzelnen Datei zusammenführen und minimieren
- [ ] CSS spät laden und Critical CSS in den Head-Bereich einfügen [6]
- [ ] Bootstrap oder ähnliche Bibliotheken sind hilfreich für eine schnelle Entwicklung, werden aber selten voll ausgenutzt. Nicht benötigte Definitionen entfernen oder Styles selber definieren. CSS Grid z.B. wird mittlerweile von allen gängigen Browsern unterstützt

4. Webfonts

- [ ] Im Idealfall Systemschriften nutzen
- [ ] Anzahl der Webfonts reduzieren und prüfen, welche tatsächlich benötigt werden
- [ ] Icon-Fonts sind oft unnötig groß. Reduzieren auf nur benötigte Icons und diese als SVG oder Sprite ausliefern (reduziert auch geladenes CSS, da zusätzlich zur Font auch eine extra CSS geladen wird)
- [ ] Webfonts können über *font subsetting* auf die benötigten Schriftzeichen reduziert werden, um bis zu 80% einzusparen[7]

---

[4] Im Idealfall werden Scripte vorher minimiert. Sollte das nicht möglich sein, können CMS dies auch übernehmen bzw. über Plugins dazu befähigt werden. In Wordpress z.B. Autoptimize
[5] https://medium.com/@netxm/what-is-tree-shaking-de7c6be5cadd
[6] über Tools und Plugins kann Critical CSS generiert und eingebunden werden z.B. über https://www.sitelocity.com/critical-path-css-generator
[7] https://walterebert.com/blog/subsetting-web-fonts

5. Caching/Hosting

- ☐ Wechsel zu einem “grünen”, nachhaltig agierenden Hoster
- ☐ *Caching* kann die Serverlast bei statischen Dateien reduzieren. Statische Dateien sollten für mindestens 1 Jahr über entsprechende Caching-Richtlinien vorgehalten werden
- ☐ Die meisten CMS können von Haus aus oder über Plugins eine statische Version der Seite vorhalten, damit Server diese nicht jedes Mal neu bearbeiten müssen. Dies könnte auch über ein Server-Caching gelöst werden. [8]
- ☐ Text-Kompression am Server (Einsatz von GZIP und idealerweise Brotli am Server konfigurieren)
- ☐ Server-Anfragen verringern/minimieren

6. Sonstiges

- ☐ *DOM*-Größe reduzieren. Onepager haben oft größere Datenmengen und dadurch längere Ladezeiten. Prüfen, ob eine Content-Aufteilung auf Unterseiten sinnvoll ist
- ☐ *SEO* nicht vernachlässigen. Inhalte, die leichter zu finden sind, verbrauchen weniger Ressourcen

7. Tools

Verschiedene Tools können eine guten Überblick darüber geben, wie Performant und nachhaltig eine Webseite ist. Einige geben auch direkt Vorschläge, welche Möglichkeiten zur Optimierung bestehen.

- Digital Beacon: Berechnet den CO_2 Fußabdruck einer Seite beim ersten und wiederholten Aufrufen. Gibt jedoch keinen Überblick über das gesamte “Gewicht”, da nicht die komplette Seite in die Berechnung mit einfließt.
- Ecograder: Ein weiteres Tool das den CO2 Fußabdruck berechnen kann
- Cabin: CO_2 bewusstes, Nachhaltiges Analytics Tool das auch den CO_2 Fußabdruck jeder Seite überprüft (Externes Tool mit Bezahlmodell)
- Google PageSpeed Insights und GTMetrix: Zeigen die aktuelle Performance einer Webseite an. Listen die Möglichkeiten für eine weitere Optimierung auf.

[8] Für Wordpress gibt es viele Plugins mit mal mehr und mal weniger Funktionen. Ein einfaches und gutes Plugin wäre z.B. WP Fastest Cache