CONTENT ENCODING (BROTLI & GZIP) UND MINIFY HTML

Raymond Eiber

Die Wichtigkeit einer guten Seitenladezeit für bessere Rankings ist mittlerweile sicherlich allseits bekannt. Oft gibt es allerdings noch kleine Stellschrauben, die man nutzen kann, um das Maximum aus einem bestehenden System herauszuholen. Man kann zum Beispiel sämtliche textuelle Assets wie JavaScript & CSS-Dateien verkleinern, indem man beispielsweise redundanten oder mittlerweile unnötigen Code entfernt. Dabei wird jedoch oft das eigentliche HTML-Dokument vergessen. Beim Content Encoding ist es in der Regel schon damit getan, dass es aktiviert ist, doch die beste Performance erreicht man auch hier mit den richtigen Einstellungen. Raymond Eiber zeigt, wie das geht.

Content Encoding im Allgemeinen

Content Encoding bietet die Möglichkeit, die Ladezeit auf allen Seiten immens zu verbessern. Hierbei wird bei einem Seitenaufruf eine komprimierte Version der eigentlichen Dateien dem Anfragenden zugeschickt. Der Browser-Client wiederum entpackt die empfangenen Daten in (Milli-)Sekundenschnelle, sodass die Seite dargestellt werden kann. Das gängigste Komprimierungsformat nennt sich GZip und wird auch von vielen Webseiten schon aktiv eingesetzt. Brotli ist ein neues Format, welches vor ca. vier Jahren von Google vorgestellt wurde.

Funktionsweise von GZip:

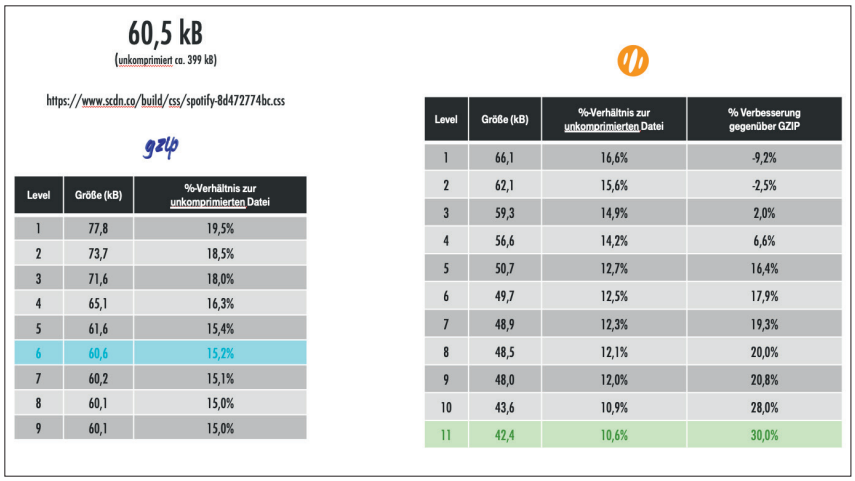
Wird GZip aktiviert, analysiert der Server die vorhandene Datei auf doppelte Datenbestandteile (zum Beispiel doppelte Zeichenketten). Diese werden mit dem Verweis auf einen bereits vorhandenen Bestandteil ersetzt. Es erfolgt somit eine Kürzung des Dokumentes, bis

Datei	Größe in kB	Content Encoding
main.css	399,0	Keine
main.css.gz	60,5	GZip

Abb.1: Normale CSS-Datei und das GZip-Äquivalent

DER AUTOR

Raymond Eiber ist technischer SEO Consultant bei der eology GmbH. Er berät KMU ebenso wie Konzerne bei der Positionierung im Netz und erarbeitet für diese langfristige SEO-Strategien.



Level	Größe (kB)	%-Verhältnis zur unkomprimierten Datei
1	77,8	19,5%
2	73,7	18,5%
3	71,6	18,0%
4	65,1	16,3%
5	61,6	15,4%
6	60,6	15,2%
7	60,2	15,1%
8	60,1	15,0%
9	60,1	15,0%

Level	Größe (kB)	%-Verhältnis zur unkomprimierten Datei	% Verbesserung gegenüber GZIP
1	66,1	16,6%	-9,2%
2	62,1	15,6%	-2,5%
3	59,3	14,9%	2,0%
4	56,6	14,2%	6,6%
5	50,7	12,7%	16,4%
6	49,7	12,5%	17,9%
7	48,9	12,3%	19,3%
8	48,5	12,1%	20,0%
9	48,0	12,0%	20,8%
10	43,6	10,9%	28,0%
11	42,4	10,6%	30,0%

Abb.2: Unterschiedliche Kompressionsstufen von GZip und Brotli

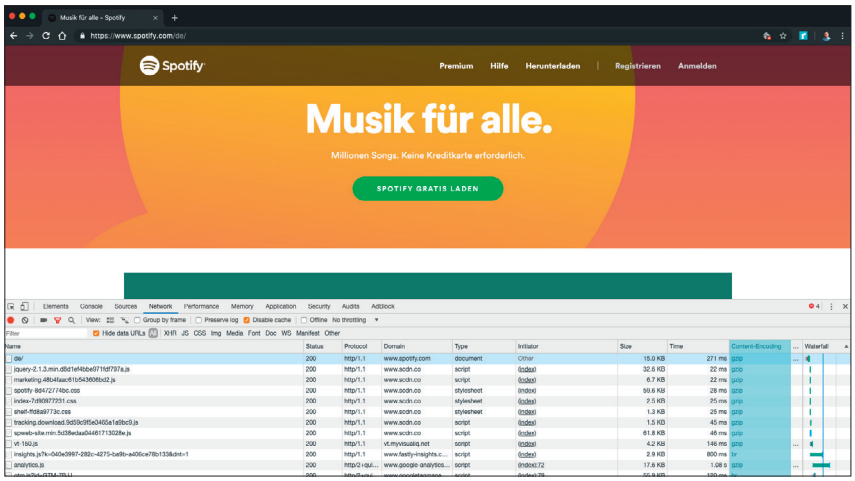


Abb.3: Content Encoding in der Entwicklerkonsole überprüfen

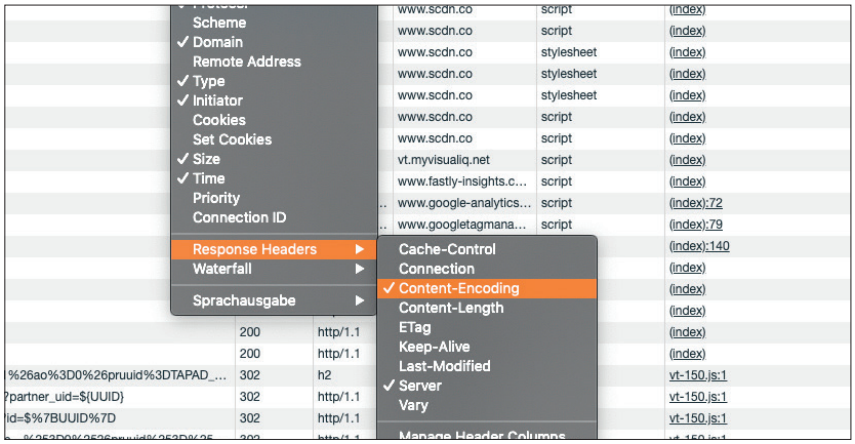


Abb.4: Einstellung von Content Encoding in der Entwicklerkonsole (Google)

es keine doppelten Verweise in einer bestimmten Größe mehr gibt. Anschließend wird die Datei im .gz-Format abgespeichert.

Dieses Thema wird in der Praxis häufig vorschnell abgeschlossen, sobald der gängigste Content-Encoding-Typ GZip auf dem Server aktiviert ist. Jedoch bleiben dann weitere Stellschrauben unbekannt und somit auch unberührt.

Statische Kompression:

Hier werden alle Assets (z. B. JavaScript & CSS) manuell in zwei Versionen auf dem Server abgelegt:

> „Durch eine falsch eingestellte Content-Encoding-Konfiguration kann es durchaus vorkommen, dass GZip performanter ist als Brotli!“

einmal die Ausgangsdatei und einmal die komprimierte Version. Erhält nun der Server eine Anfrage für eine Datei, so sucht er anhand der Datei-Endung zuerst, ob es diese Datei in der komprimierten Version gibt.

Findet der Server keine komprimierte Datei, so spielt dieser einfach die normale Datei aus. Vorteil hierbei ist, dass die Daten sich schon in der kompakten Version auf den Server befinden und nicht erst generiert werden müssen.

Dynamische Kompression:

Anders als bei der statischen Kompression wird jede angefragte textuelle Datei während der Anfrage komprimiert und anschließend ausgeliefert. Hierbei lassen sich jedoch auch weitere Einstellungen vornehmen. Zum Beispiel: Ist eine Datei zu klein, muss der Server diese nicht komprimieren. Dies würde nämlich unnötig die Zeit verlängern, bis die Datei ausgeliefert werden kann.

Kompressionsstufe:

Dies gehört zu den wichtigsten Einstellungen vom Content Encoding. Hier wird nämlich die Einstellung hinterlegt, wie stark komprimiert werden soll. Geht man hier auf die maximale Einstellung, so muss der Server dementsprechend auch mehr arbeiten (bei

einer dynamischen Kompression), was zu einer längeren Auslieferungszeit der Datei führen kann. Sofern der Server die zusätzliche Last tragen kann (was bei einer modernen Server-Architektur der Fall sein sollte), besteht hier die Möglichkeit, durch eine Einstellung die Dateigröße signifikant zu verkleinern.

TIPP

Möchte man einen neuen Content-Encoding-Typ auf einem Shared Host aktivieren, so reicht meist ein freundlicher Anruf beim Hostanbieter.

Oft weiß jedoch der SEO-Verantwortliche nicht genau, auf welche Komprimierungsstufe sein System aktuell eingestellt ist. Hier kann die Seite *tools.paulcalvano.com/compression.php* Abhilfe schaffen. Dieses kostenfreie Tool nimmt die eingetragene Datei und komprimiert diese sowohl auf jeder Kompressionsstufe von GZip als auch von Brotli. Als Ergebnis werden eine geschätzte Stufe und der jeweilige Content-Encoding-Typ kommuniziert. Wie in Abbildung 2 verdeutlicht, ist auf den Servern von Spotify GZip mit einer Kompressionsstufe von 6 eingestellt. Würde man hier auf Brotli mit der maximalen Kompressionskraft umstellen, so würde die Datei um rund 30 % verkleinert werden.

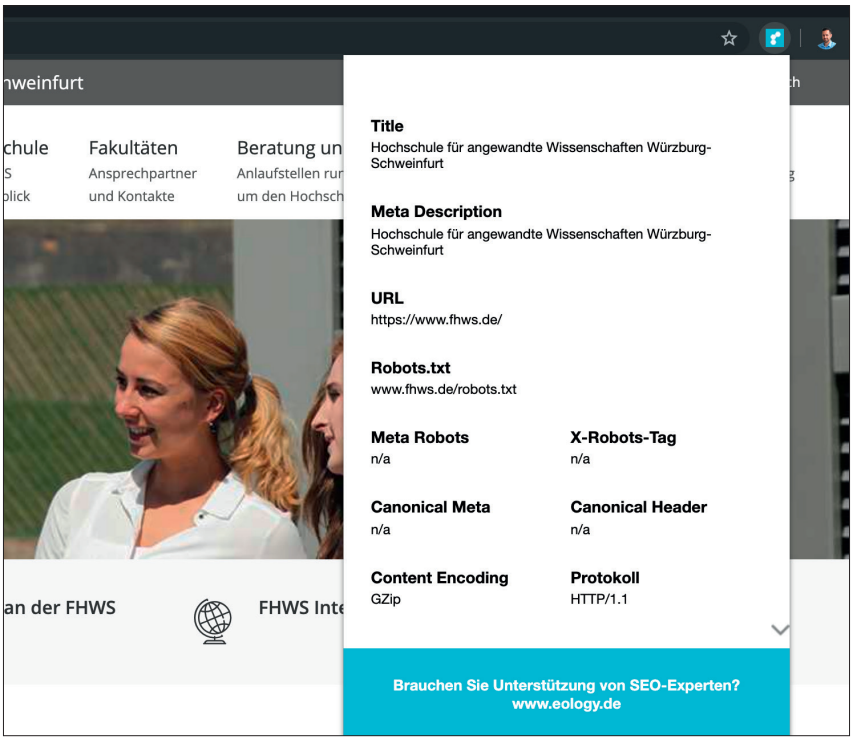


Abb.5: Google-Chrome-Plug-in für Content Encoding

	GZip	Brotli
Namenherkunft	GNU Zip	Schweizer Gebäck
Dateiendung	*.gz	*.br
Entwickler	Jean-Loup Gailly, Mark Adler	Google
Erscheinungsjahr	1992	2015
Kann Bilder komprimieren	Nein	Ja
Komprimierungsstufen	1-9	1-11
Besonderheit	» Frei von patentierten Algorithmen » Sehr schnell im Komprimieren im Vergleich zu Brotli	» Benutzt ein vordefiniertes Wörterbuch mit ca. 13.000 Einträgen der meistgenutzten Ausdrücke » Komprimiert bis zu 30 % stärker als GZip
HTTPS nötig?	Nein	Ja

Abb.7: Vergleich GZip und Brotli

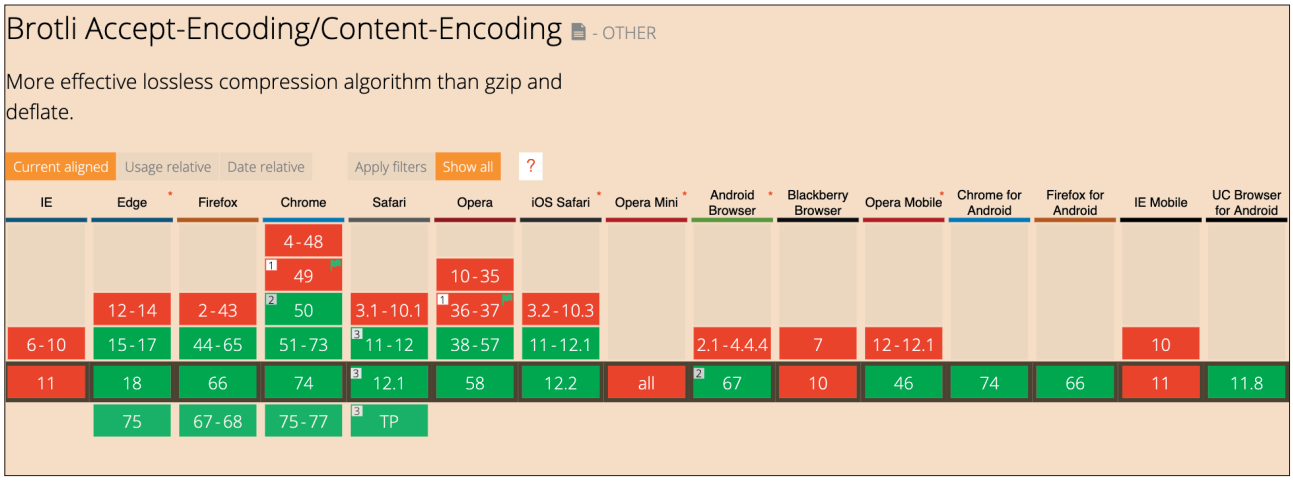


Abb.6: Browserunterstützung für Brotli

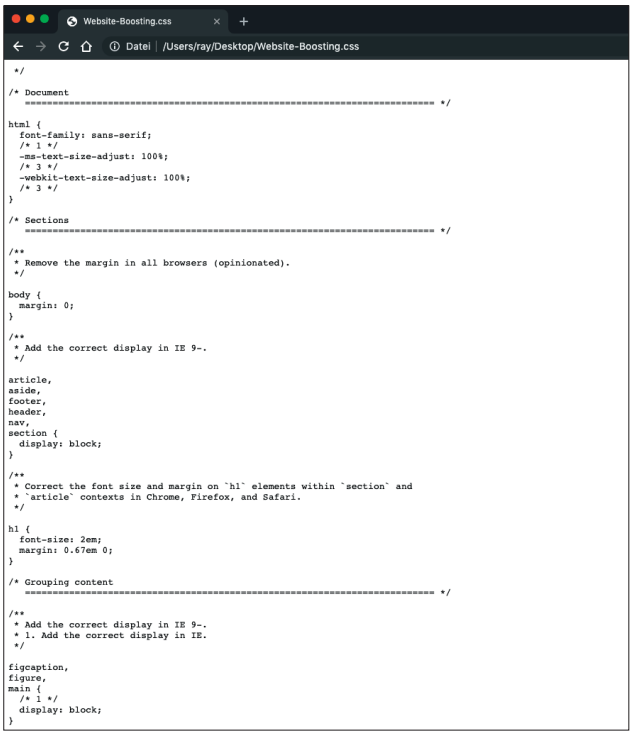


Abb.8: Normale CSS Datei

Abb.9: „Minifizierte“ CSS Datei

Möchte man generell überprüfen, welcher Content-Encoding-Typ aktuell eingestellt ist, so ist dies leicht mit der Entwicklerkonsole von Google Chrome (oder Browser mit Chromium wie Opera, Brave oder bald auch Microsoft Edge) zu überprüfen.

Hierzu muss man wie folgt vorgehen:

1. Entwicklerkonsole öffnen
   Windows: [F12] oder [Strg] + [Umschalt] + [J]
   MacOS: [⌘]+[Optionstaste]+[J]
2. Reiter Netzwerk öffnen
3. In der Tabellenleiste Rechtsklick und unter Response Headers das Feld Content Encoding auswählen (siehe Abbildung 4)
4. Seite laden oder aktualisieren

Möchte man sich nur einen schnellen Überblick über den eingesetzten Content-Encoding-Typ verschaffen, so kann man z. B. auch das Google-Chrome-Browser-Plug-in „eology site metrics“ (*http://einfach.st/eoplugin*) verwenden.

Da es heute schon mehrere Typen von Content Encoding gibt, kann man davon ausgehen, dass in Zukunft noch mehr Möglichkeiten geschaffen werden. Browser müssen den jeweiligen Encoding-Typ unterstützen, um die Daten wieder zu dekomprimieren. Um sich einen Überblick der Browser zu verschaffen, welche einen bestimmten Content-Encoding-Typ unterstützen, lohnt sich ein Blick auf die Seite caniuse.com.

Wird auf einer Webseite ein CDN zwischengeschaltet, so gilt auch hier zu überprüfen, welche Komprimierung mit welcher Stärke eingesetzt wird. Standardmäßig ist auf jedem CDN mindestens GZip aktiviert.

Zopfli: GZip auf Steroiden:

Ist es für ein Unternehmen wirtschaftlich nicht sinnvoll, auf Brotli umzusteigen, so gibt es die Möglichkeit, die Kompressionsstufe zu erhöhen. Ist diese schon ausgereizt und man sehnt sich nach noch mehr Performance mit GZip, so kann Zopfli Abhilfe schaffen.

Zopfli ist ein effizienter Komprimierungsalgorithmus, welcher GZip-Dateien generieren kann. Vorteil hierbei ist, dass jeder Client die Dateien entpacken kann, ohne dass der Browser eine neue Technologie erlernen muss. Nachteil ist jedoch, dass Zopfli viel länger braucht, um die Daten zu verarbeiten. Der beste Ansatz ist, Zopfli als statische Komprimierung für Daten zu verwenden, die nicht während einer Abfrage komprimiert werden müssen.

Anhand einiger Benchmark-Ergebnissen wird die Dateigröße zwischen 4 % und 8 % reduziert (*http://einfach.st/compr4*).

Interessant ist zudem, dass sowohl Zopfli als auch Brotli in der hauseignen Google-Code-Bäckerei konzipiert und entwickelt wurden.

Recommendations

You should minify your HTML, CSS, and JavaScript resources:

- **To minify HTML,** try HTMLMinifier
- **To minify CSS,** try CSSNano and csso.
- **To minify JavaScript,** try UglifyJS. The Closure Compiler is also very effective. You can create a build process that uses these tools to minify and rename the development files and save them to a production directory.

Abb.10: Google Developer; Empfehlung, Ressourcen zu minifizieren

Minify im Allgemeinen

Beim Programmieren einer Webseite erschafft man sich eine schöne Struktur mit Zeilenabständen und Umbrüchen der geschriebenen Code-Zeilen. Dies ist dann für jeden Programmierer angenehm zu lesen und bewahrt somit auch die Übersicht für eine Wartung bzw. eine Überarbeitung der Datei. Die Maschine jedoch (in dem Fall der Browser), braucht keine Verschönerung der Datei, sondern benötigt lediglich den relevanten Code. All diese unnötigen Zeichen wie Umbrüche, Tabstopps etc. fallen zur Last der Größe der Datei. Daher ist es ratsam, die Datei, welche für den Livebetrieb einer Website genutzt wird, zu minifizieren. Um hier zu überprüfen, ob aktuell die Dateien minifiziert werden, muss man einfach die einzelne Datei im Browser öffnen. Sieht man auf den ersten Blick viel Whitespace (weiße Fläche), so signalisiert dies, dass der Code nicht optimiert wurde.

Google selbst empfiehlt diese Art der Pagespeed- Optimierung und kommuniziert dies auf der eigenen Entwicklerplattform; siehe *http://einfach.st/minify*.

Nach einer Minifizierung der jeweiligen Ressourcen sollte man jedoch unbedingt die Webseite vor dem Live-Gang testen. Probleme kann es zum Beispiel beim Entfernen relevanter Kommentare im HTML-Dokument geben. Hier sollten nicht die Kommentare entfernt werden, welche den Browser erkennen lassen, ob es sich aktuell um einen Internet-Explorer-Browser handelt.

Einen großen Vorteil gibt es noch zusätzlich bei JavaScript-Dateien. Hier können ergänzend zum Minifizieren noch die Namen der Funktionen sowie Variablen durch Buchstaben ersetzt werden. Des Weiteren gibt es die Option, alle Variablen am Anfang der JavaScript-Datei als Array zu definieren. Hierdurch wird die Definition einer

```
<!--[if IE]>
Place content here to target all Internet Explorer users.
<![endif]-->
```

Abb.11: Internet-Explorer-Kommentare

```
// JavaScript Code
function add(firstNumber, secondNumber) {
    return firstNumber + secondNumber;
};

// JavaScript Code + Minify inkl. Variable Renaming
function a(b,c){return b+c};
```

Abb.12: Minifiziertes JavaScript + Variable Renaming

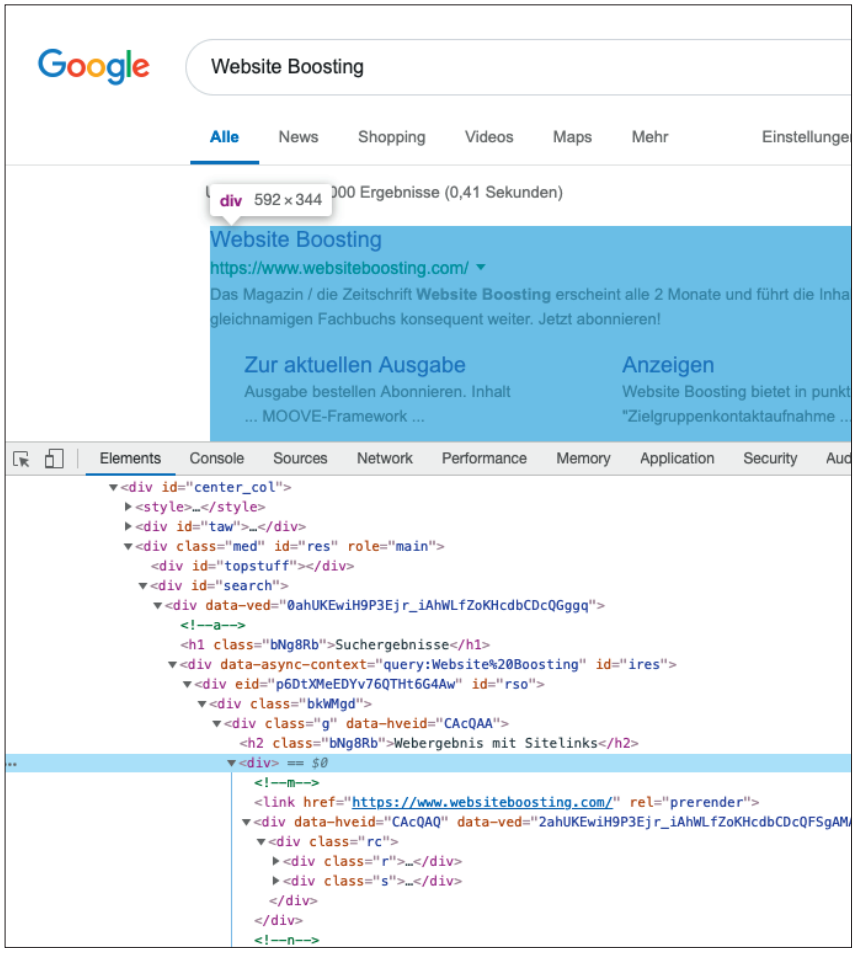


Abb.13: Google kurze Klassennamen

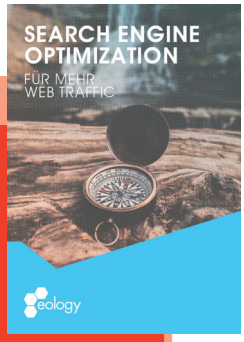
	Normal (in kB)	Minify (in kB)	GZip		Brotli	
			Normal	Minify	Normal	Minify
Amazon.de	429,3	351,6	108,0	88,3	86,0	70,0
Wetter.com	511,4	423,1	86,3	82,9	66,1	63,9
BAUR.de	186,4	180,0	33,6	32,1	26,5	25,6
Spiegel.de	362,4	335,1	73,4	69,6	57,6	55,1
ImmobilienScout24.de	334,8	323,7	56,6	55,4	45,8	44,8

Abb.14: Tabelle Vergleich von HTML Minify und Content Encoding

	GZip + Minify	Brotli + Minify
Amazon.de	22 %	23 %
Wetter.com	4 %	3 %
BAUR.de	5 %	4 %
Spiegel.de	5 %	5 %
ImmobilienScout24.de	2 %	2 %
Ergebnis	**8 %**	**7 %**

Abb.15: Tabelle Prozentersparnis von HTML Minify und Content Encoding

TIPP



Weitere Tipps zum Optimieren des Pagespeeds finden Sie im kostenlosen Whitepaper „Search Engine Optimization für mehr Web Traffic“ unter *http://einfach.st/eologypaper3.*

Variablen „var“ im Dokument gespart.

Möchte man wirklich das Maximum aus der Minifizierung herauskitzeln, so kann man auch wie Google allen Klassennamen und weiteren Elementen im HTML-Dokument kurze Namen geben.

Abschließend ist zur Thematik Minify noch zu sagen, dass diese Methodik als lossy (Englisch für „mit Verlust“) deklariert wird. Dies bedeutet, dass Daten entfernt werden und diese nicht 1:1 wieder in ihre ursprüngliche Form gebracht werden können. Beim Content Encoding kann man immer wieder durch die Dekompression das Original-Dokument herstellen. Diese Methode nennt sich lossless (Englisch für „ohne Verlust).

Minify + HTML

Bei Programmierern stellt sich nun (zu Recht) die Frage, ob es sich denn lohne, eine HTML-Datei zu minifizieren, wenn diese nach einem ähnlichen Schema schon über GZip oder Brotli komprimiert wurde. Wie oben bei der Funktionsweise von GZip erklärt, werden doppelte Zeichenfolgen erkannt und auf den ersten Fund jener Zeichenkette verwiesen. Hierzu wurde ein Test aufgesetzt, bei dem HTML-Dokumente folgender Webseiten genommen wurden, um zu prüfen, ob die Kombination von HTML-Minifizieren mit Content Encoding die Dateigröße positiv beeinflusst.

Wie in der Abbildung 14 zu sehen ist, wird durch die Kombination beider Optimierungsmaßnahmen ein positives Ergebnis erzeugt.

Bei diesem Test wurde jeweils mit der stärksten Komprimierungsstufe von GZip (Stufe 9) und Brotli (Stufe 11) gearbeitet.

In diesem Rahmen erfolgte beim Minifizieren eine Änderung folgender Elemente:

- Entfernung Whitespace (Zeilenumbrüche, Tabstopps, Leerzeichen etc.)
- Entfernung Kommentare (außer funktionale Kommentare)
- Entfernung redundanter Attribute (leere Attribute wie style=““)
- Entfernung ‹script› Attribute (z. B. “text/javascript”)
- Kurzschreibweise für Doctype-Benennung

Die Tabelle in Abbildung 15 zeigt deutlich, dass es eine Dateiverkleinerung von ca. 3-7 % erzeugen könnte, auch wenn Amazon.de hier die Ergebnisse stark beeinflusst.

Fazit

Wurden die größten Hebel schon umgesetzt, welche die Seitenladezeit einer Webseite verbessern könnten, so muss man anfangen, komplexere technische Maßnahmen anzuwenden. Weitere Einstellungsmöglichkeiten, wie die Kompressionsstärke auszureizen oder auch das eigentliche HTML-Dokument zu minifizieren, können einen weiteren Pagespeed-Boost erzeugen, wofür die Besucher dankbar sein werden. Und auch wenn in der Abbildung 14 nur wenige kB eingespart wurden, darf nicht vergessen werden, dass diese kB pro Nutzer pro Seitenaufruf gelten (bei den HTML-Dokumenten). Nimmt man die Zahlen der Seitenaufrufe aus seinem Analysetool und rechnet die ersparten kB hoch, so erkennt man schnell, wie viel Datenbandbreite eigentlich gespart wurde. ¶