Schon gewusst? Content-Encoding!

Raymond Eiber

Viele haben schon oft Begriffe wie „gzip“ und „Brotli“ gehört. Noch mehr Leute wissen, dass diese Komprimierungen gut für die Seitenladezeit sind. Doch was steckt bei den Themen wirklich dahinter? Was sind die Kompressionsstufen oder was bedeutet dynamische und statische Komprimierung? All diese Deep Dive Insights erfahren Sie heute in diesem Artikel

Als das Internet das Licht der Welt erblickte, kam gleichzeitig das Thema Seitenladezeit mit auf. Wer möchte denn schon lange vor dem Bildschirm warten, bis was passiert – keiner. Zu Beginn des WWW gab es überwiegend nur textuelle Assets (HTML, JS & CSS), da die Infrastruktur für das Internet noch nicht ausgebaut war und Bilder sehr lange laden mussten. Ein großer Meilenstein hinsichtlich der Seitenladezeit wurde anschließend im Jahr 1992 erreicht, als die textuelle Komprimierung gzip publiziert wurde; Brotli ist ein weiteres Format zur Datenkomprimierung, das von Google entwickelt und 2015 erstmals im Google Open Source Blog annonciert wurde.

Wie funktioniert die Komprimierung?

Funktionsweise gzip

gzip ist ein Programm zur Dateikomprimierung und -dekomprimierung, das den DEFLATE-Algorithmus verwendet. Der DEFLATE-Algorithmus ist eine Kombination aus LZ77 (einer Methode, die redundante Daten identifiziert und durch kürzere Referenzen ersetzt) und Huffman-Codierung (eine Methode, die häufig vorkommende Datenmuster durch kürzere Codes ersetzt).

Hier sind die grundlegenden Schritte, wie gzip funktioniert:

1. Redundanzerkennung mit LZ77: Das Programm analysiert die Daten und sucht nach wiederholten Zeichenketten. Wenn es solche Zeichenketten findet, ersetzt es sie durch kürzere Referenzen, die auf die ursprüngliche Position und Länge der Zeichenkette hinweisen.
2. Huffman-Codierung: Nach der LZ77-Kompression wird die Datenfolge weiter komprimiert, indem häufig vorkommende Muster durch kürzere binäre Codes und seltene Muster durch längere Codes dargestellt werden.
3. Dateiheader hinzufügen: Schließlich fügt gzip einen kleinen Header an den Anfang der komprimierten Datei hinzu, der Informationen wie den ursprünglichen Dateinamen, das Datum der Modifikation und andere Metadaten enthält.

Bei der Dekompression wird dieser Prozess im Wesentlichen in umgekehrter Reihenfolge durchgeführt. Das gzip-Programm liest den Header, um zu erfahren, wie die Daten dekomprimiert werden sollen. Dann wendet es die Huffman-Decodierung und die LZ77-Decodierung an, um die ursprüngliche Datei wiederherzustellen.

Kennung einer gzip-Datei: .gz

DER AUTOR

Raymond Eiber ist Expert SEO Consultant bei diva-e in München. Er betreut diverse Kunden im Enterprise-Umfeld und entwickelt interne SEO-Tools, wie zum Beispiel das diva-e SEO-Plug-in.

Foto: Mykola Lishchyshyn / gettyimages.de

Funktionsweise Brotli

Brotli ist ein Kompressionsalgorithmus, ähnlich wie DEFLATE (der von gzip verwendet wird). Dieser Algorithmus wurde von Google entwickelt und hat einige Unterschiede zu gzip, die ihn in vielen Fällen effizienter machen. Ursprünglich wurde Brotli entwickelt, um Webfonts schneller laden zu können.

Im Gegensatz zu gzip verwendet Brotli keinen festen Satz von Huffman-Tabellen. Brotli erstellt mehrere Tabellen und verwendet die, die in einem bestimmten Kontext am besten passt. Das bedeutet, dass Brotli die Art und Weise, wie es Daten komprimiert, an die spezifische Natur der Daten anpasst, was zu einer besseren Kompression führt.

Kennung einer Brotli-Datei: .br

Was ist der Unterschied zwischen dynamischer und statischer Komprimierung?

Bei der dynamischen Komprimierung werden die Antworten vom Server automatisch komprimiert, sobald diese angefragt werden. Das heißt, bevor die Daten an den Browser rausgehen, wird ein Prozess angestoßen, sodass die zu sendenden Daten „zusammengepackt" und dann erst verschickt werden.

Klassische Use-Cases hierfür sind zum Beispiel API-Abfragen, da diese oft unterschiedliche bzw. dynamische Inhalte erzeugen. Zudem können API-Abfragen wie zum Beispiel ein JSON-Format oder ein XML-Format sehr gut komprimiert werden. Ursachen sind wiederkehrende Schlüssel, Leerzeichen und geschweifte Klammern.

```
brotli on;
brotli_comp_level 6;
brotli_static off;
brotli_types application/atom+xml application/javascript application/json application/rss+xml
             application/vnd.ms-fontobject application/x-font-opentype application/x-font-truetype
             application/x-font-ttf application/x-javascript application/xhtml+xml application/xml
             font/eot font/opentype font/otf font/truetype image/svg+xml image/vnd.microsoft.icon
             image/x-icon image/x-win-bitmap text/css text/javascript text/plain text/xml;
  return go(f, seed, [])
}
```

Abb. 1: nginx-Konfigurationsdatei für Brotli mit der Einstellung einer dynamischen Komprimierung

```
// webpack.mix.js
const mix = require('laravel-mix');
require('laravel-mix-compress');

mix.js('src/app.js', 'js');
mix.css('src/app.css', 'css');
mix.compress({
    productionOnly: false,
    minRatio: 1,
    useBrotli: true
});
```

Abb. 2: Konfigurationsdatei von einem Module-Bundler mit aktivierter Brotli-Komprimierung

Verwendet man eine statische Komprimierung, so wird beim s. g. build („die Erstellung") der CSS- und JavaScript-Dateien ein weiteres komprimiertes Format auf dem Server abgelegt (zum Beispiel style.css.gz). Da die komprimierten Dateien schon im Vorfeld auf dem Server liegen, verkürzt sich die Antwortzeit des Servers.

In Abildung 2 sehen Sie Beispiel von einem Module-Bundler namens Laravel Mix (aufbauend auf dem Framework webpack).

Mit welcher Komprimierungsstufe sollte komprimiert werden?

Die Kompressionsstufe bestimmt, wie stark ein Datensatz komprimiert wird. Bei Formaten wie gzip oder Brotli gibt diese Stufe an, welches Gleichgewicht zwischen Kompressionszeit und Kompressionsrate gewählt wird. Höhere Stufen führen in der Regel zu stärkerer Komprimierung, benötigen aber mehr Zeit und Ressourcen (CPU-Leistung).

gzip bietet neun Komprimierungsstufen, die mit den Zahlen 1 bis 9 angegeben werden:

» 1 oder fast: schnellste Kompression, aber nicht die beste Kompressionsrate.
» 9 oder best: langsamste Kompression, aber die beste Kompressionsrate.

Die Zahlen dazwischen ermöglichen unterschiedliche Kompromisse zwischen Geschwindigkeit und Kompressionsrate. In vielen Anwendungen wird oft der Standardwert (Stufe 6) verwendet, da er eine ausgewogene Kombination aus Geschwindigkeit und Kompression bietet.

Brotli hat ebenfalls verschiedene Komprimierungsstufen, wobei es insgesamt elf Stufen gibt:

» 1: schnellste Kompression, aber mit der niedrigsten Rate.
» 11: langsamste Kompression, aber mit der höchsten Rate.

Wiederum ermöglichen die Zahlen dazwischen verschiedene Kompromisse zwischen Geschwindigkeit und Kompressionsrate.

TIPP

Hier können Sie alle Brotli-Einstellungen für einen nginx- und einen Apache-Server sehen:
» https://docs.nginx.com/nginx/admin-guide/dynamic-modules/brotli/
» https://httpd.apache.org/docs/2.4/mod/mod_brotli.html

TIPP

Mit diesem SEO-Plug-in (für Chrome Browser) können Sie die aktuelle Komprimierung sehen und die geschätzte Komprimierungsstufe überprüfen: *einfach.st/plugin532*.

Bei einem Komprimierungs-Check der größten JavaScript-Datei von *www.websiteboosting.com* ist ersichtlich, dass bei einer stärkeren Komprimierung viele Kilobit eingespart werden könnten.

Mit maximaler Brotli-Kompression könnten zum Beispiel weitere 15 % Daten eingespart werden.

Möchten Sie selbst Ihre Komprimierung testen, so können Sie dies hier tun: *einfach.st/seopl5*.

Welche Daten sollten überhaupt komprimiert werden?

Es ist generell möglich, alle Datentypen, wie zum Beispiel Bilder und Videos, zu komprimieren. Jedoch führen diese selten zu einer Datenersparnis (die Daten können zum Teil danach auch noch größer sein und es dauert noch länger, diese wieder zu entpacken). Textuelle Daten gehören zu den gängigen Inhaltstypen einer Website, und diese sollten komprimiert werden:

Bei diesen Daten ergibt das Sinn:

- text/html
- text/css
- text/plain
- text/xml
- text/x-component
- text/javascript
- application/x-javascript
- application/javascript
- application/json
- application/manifest+json
- application/vnd.api+json
- application/xml
- application/xhtml+xml
- application/rss+xml
- application/atom+xml

Komprimierung	Art	Stufe	Datenmenge
Unkomprimiert	-	-	626,2 kB
Aktuelle Komprimierung	gzip	6	168,7 kB
Stärkste Komprimierung	brotli	11	144,1 kB

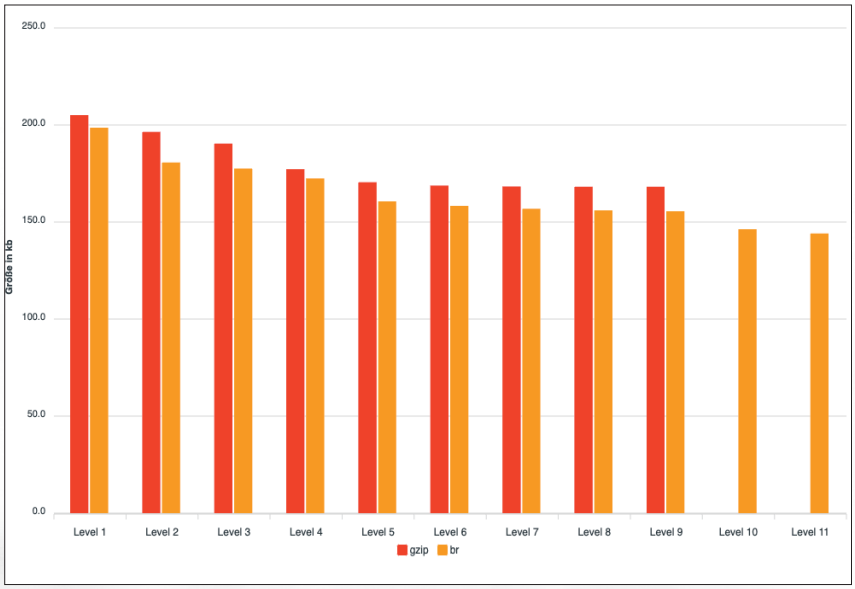

Abb. 3: Komprimierungs-Check von einer JS-Datei der Website Boosting

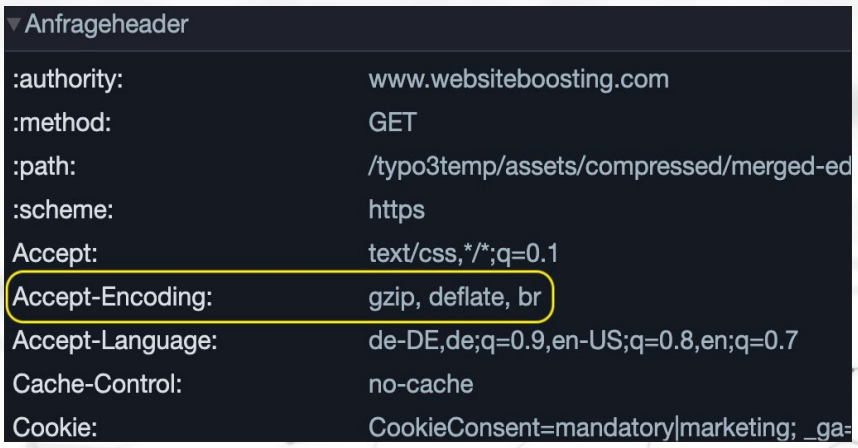

Abb. 4: Anfrage-Header

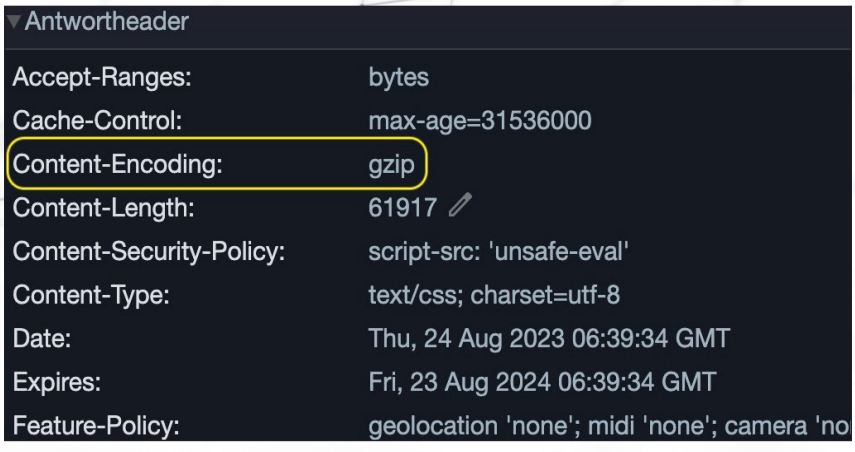

Abb. 5: Antwort-Header

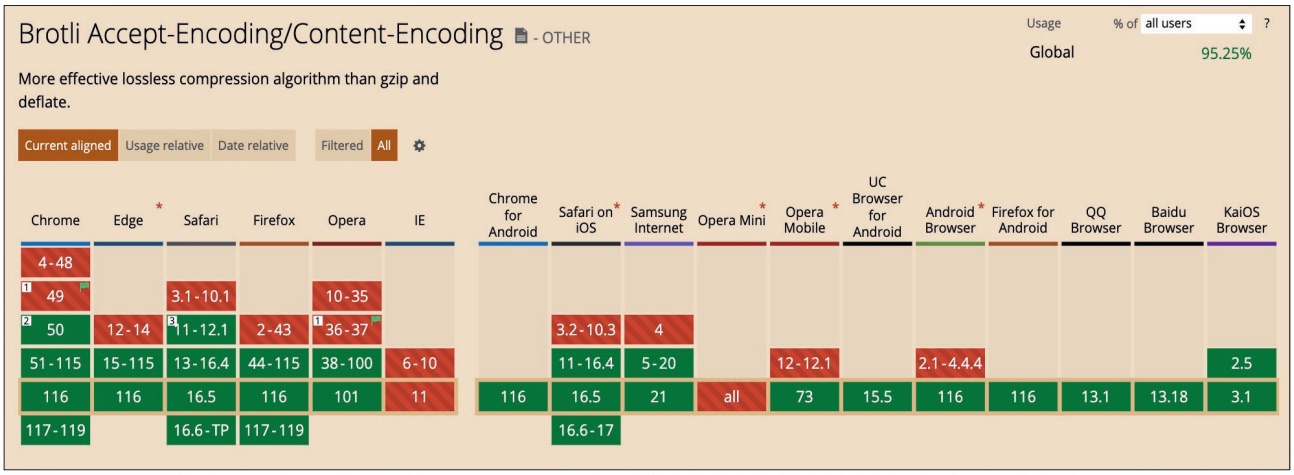

Abb. 6: Verwendung von Brotli in Webbrowsern

- application/vnd.ms-fontobject
- application/x-font-ttf
- application/x-font-opentype
- application/x-font-truetype
- image/svg+xml
- image/x-icon
- image/vnd.microsoft.icon
- font/ttf
- font/eot
- font/otf
- font/opentype

Wann wird eine Antwort komprimiert?

Wenn ein Webbrowser eine Ressource anfragt, dann gibt dieser gleich die Information mit, welche Komprimierungsformate er versteht.

Webbrowser – Accept-Encoding Header: Dieser Header wird vom Webbrowser (Client) in der Anfrage an den Server gesendet. Er informiert den Server darüber, welche Kompressionsverfahren der Browser versteht und akzeptiert. Wenn der Header zum Beispiel den Wert „Accept-Encoding: gzip, br“ hat, signalisiert dies dem Server, dass sowohl gzip- als auch Brotli-Kompressionen unterstützt werden.

Wenn eine Anfrage keinen Accept-Encoding Header enthält, wird keine komprimierte Datei versendet. Enthält diese eine Wildcard (*),darf jede Art einer Komprimierung zurückgesendet werden.

Server – Content-Encoding Header: Dieser HTTP-Header gibt an, welche Kodierungstransformationen auf den Dateninhalt zur Anwendung kamen, bevor sie an den Client gesendet wurden.

Welche Browser unterstützen welches Format?

Wichtig zu wissen ist auch, welche Browser überhaupt mit welchem Komprimierungsformat umgehen können. Dadurch, dass gzip schon 1992 veröffentlicht wurde, gibt es heute keinen modernen Browser, der nicht mit diesem Dateiformat umgehen kann. Bei Brotli gibt es noch vereinzelt einige Browser, die noch nicht mit dem Format umgehen können.

Content-Encryption in Bezug auf Online-Marketing

Das Besondere an Content-Encryption ist, dass es eine Optimierung ist, die Auswirkungen auf jede einzelne Seite hat. Das heißt, dass bei jeder Anfrage an den Server nach einer Optimierung Unmengen an Daten gespart werden können.

Direkte Vorteile von einer optimierten Kompression:

1. **Schnellere Ladezeiten**
   Die wohl größte und offensichtlichste Verbesserung ist die reduzierte Ladezeit von Webseiten.
2. **Energieeinsparung**
   Dies mag auf den ersten Blick nicht offensichtlich sein, aber die Übertragung kleinerer Datenmengen kann auch den Energieverbrauch von Servern und Client-Geräten reduzieren.
3. **Reduzierte Serverlast**
   Weniger Daten zu senden, bedeutet, dass Server weniger Arbeit leisten müssen, um denselben Inhalt zu liefern. Dies kann zu einer höheren Server-Performance und geringeren Hosting-Kosten führen.

Indirekte Vorteile einer optimierten Kompression:

1. **Verbesserung des SEO-Rankings**
   Die Geschwindigkeit (bzw. die Core Web Vitals) einer Webseite ist ein bekannter Faktor für das Ranking in Suchmaschinen wie Google.
2. **Erhöhte Benutzerzufriedenheit**
   Ein schnelles Web-Erlebnis ohne lange Ladezeiten erhöht die Zufriedenheit der Besucher, was wiederum die Absprungrate reduzieren und die Verweildauer auf der Seite erhöhen kann.
3. **Höhere Conversion-Raten**
   Schnelle Webseiten können zu höheren Conversion-Raten führen, da weniger Benutzer wahrscheinlich den Prozess aufgrund von Verzögerungen oder langsamen Ladezeiten abbrechen.