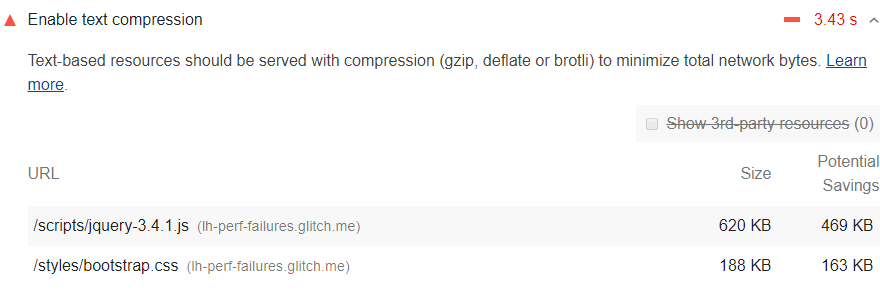
# Enable text compression

May 2, 2019 • Updated Jun 4, 2020

Appears in: [Performance audits](https://web.dev/lighthouse-performance)

Text-based resources should be served with compression to minimize total network bytes. The Opportunities section of your Lighthouse report lists all text-based resources that aren't compressed:



## How Lighthouse handles text compression [#](https://web.dev/uses-text-compression/?utm_source=lighthouse&utm_medium=devtools#how-lighthouse-handles-text-compression)

Lighthouse gathers all responses that:

* Have text-based resource types.
* Do not include a content-encoding header set to br, gzip, or deflate.

Lighthouse then compresses each of these with [GZIP](https://www.gnu.org/software/gzip/) to compute the potential savings.

If the original size of a response is less than 1.4KiB, or if the potential compression savings is less than 10% of the original size, then Lighthouse does not flag that response in the results.

The potential savings that Lighthouse lists are the potential savings when the response is encoded with GZIP. If Brotli is used, even more savings are possible.

## How to enable text compression on your server [#](https://web.dev/uses-text-compression/?utm_source=lighthouse&utm_medium=devtools#how-to-enable-text-compression-on-your-server)

Enable text compression on the server(s) that served these responses in order to pass this audit.

When a browser requests a resource, it will use the [Accept-Encoding](https://developer.mozilla.org/docs/Web/HTTP/Headers/Accept-Encoding) HTTP request header to indicate what compression algorithms it supports.

Accept-Encoding: gzip, compress, br

If the browser supports [Brotli](https://opensource.googleblog.com/2015/09/introducing-brotli-new-compression.html) (br) you should use Brotli because it can reduce the file size of the resources more than the other compression algorithms. Search for how to enable Brotli compression in <X>, where <X> is the name of your server. As of June 2020 Brotli is supported in all major browsers except Internet Explorer, desktop Safari, and Safari on iOS. See [Browser compatibility](https://developer.mozilla.org/docs/Web/HTTP/Headers/Content-Encoding#Browser_compatibility) for updates.

Use GZIP as a fallback to Brotli. GZIP is supported in all major browsers, but is less efficient than Brotli. See [Server Configs](https://github.com/h5bp/server-configs) for examples.

Your server should return the [Content-Encoding](https://developer.mozilla.org/docs/Web/HTTP/Headers/Content-Encoding) HTTP response header to indicate what compression algorithm it used.

Content-Encoding: br

## Check if a response was compressed in Chrome DevTools [#](https://web.dev/uses-text-compression/?utm_source=lighthouse&utm_medium=devtools#check-if-a-response-was-compressed-in-chrome-devtools)

To check if a server compressed a response:

1. Press `Control+Shift+J` (or `Command+Option+J` on Mac) to open DevTools.
2. Click the **Network** tab.
3. Click the request that caused the response you're interested in.
4. Click the **Headers** tab.
5. Check the content-encoding header in the **Response Headers** section.

The content-encoding response header.

To compare the compressed and de-compressed sizes of a response:

1. Press `Control+Shift+J` (or `Command+Option+J` on Mac) to open DevTools.
2. Click the **Network** tab.
3. Enable large request rows. See [Use large request rows](https://developers.google.com/web/tools/chrome-devtools/network/reference#request-rows).
4. Look at the **Size** column for the response you're interested in. The top value is the compressed size. The bottom value is the de-compressed size.

Gzip Compression reduces the size of HTML files, JavaScripts, CSS stylesheets and XML files. Generally, Enable(ing) **Gzip Compression in WordPress**with plugin reduces 60% to 80% size of your site. But I am saving 35,060 bytes using Gzip Compression in WordPress via .htaccess for *technumero.com*. And it is reducing webpage size by 82.2%. In this article, I will discuss what is Gzip Compression, how does it work, how to enable Gzip Compression in WordPress.

**Gzip Compression can be enabled with or without using a WordPress plugin. You can easily enable Gzip Compression by adding the given below code in your .htaccess file. The .htaccess file can be found on your server (Apache) in the root directory of your domain.**

Enabling Gzip Compression in WordPress via .htaccess reduces Server Response Time and volume of data sent by the server to a web browser. Moreover, compressed page size help reducing the transferred response and data.

The best way to enable compression is, using **mod\_gzip** or **mod\_deflate** in WordPress **.htaccess file**. And the good news is, Gzip compression can be enabled by adding a simple code in WordPress .htaccess file (without plugin method) of your website.

Don’t worry if you don’t want to enable compression via [.htaccess file](https://technumero.com/increase-page-speed-using-htaccess-wordpress/).  I have also explained how to **enable Gzip compression in WordPress using plugins** like [W3 Total Cache](https://technumero.com/install-configure-w3-total-cache-wordpress-plugin/), WP Super Cache, and WP Fastest Cache, etc. In addition to that, I will show you how to enable DEFLATE compression.

Gzip Compression in WordPress

**Don’t be confused with the title** – the Gzip and DEFLATE compression methods using .htaccess file and configuration file are server-dependent (i.e. Apache and Nginx). That means these **methods work well** with **non-WordPress sites like HTML and other CMS based websites etc**.

Table of Contents [[hide](https://technumero.com/gzip-compression-in-wordpress/)]

* [What is Gzip Compression?](https://technumero.com/gzip-compression-in-wordpress/#What_is_Gzip_Compression)
* [How does it work?](https://technumero.com/gzip-compression-in-wordpress/#How_does_it_work)
* [DEFLATE Compression vs Gzip Compression: The Difference](https://technumero.com/gzip-compression-in-wordpress/#DEFLATE_Compression_vs_Gzip_Compression_The_Difference)
* [How to Check Gzip Compression](https://technumero.com/gzip-compression-in-wordpress/#How_to_Check_Gzip_Compression)
* [Enabling Gzip & DEFLATE Compression in WordPress](https://technumero.com/gzip-compression-in-wordpress/#Enabling_Gzip_DEFLATE_Compression_in_WordPress)
  + [Enable Gzip Compression in WordPress via .htaccess](https://technumero.com/gzip-compression-in-wordpress/#Enable_Gzip_Compression_in_WordPress_via_htaccess)
  + [Enable Gzip Compression on Apache server](https://technumero.com/gzip-compression-in-wordpress/#Enable_Gzip_Compression_on_Apache_server)
  + [Enable DEFLATE Compression in WordPress](https://technumero.com/gzip-compression-in-wordpress/#Enable_DEFLATE_Compression_in_WordPress)
  + [Enable Gzip Compression on Nginx server](https://technumero.com/gzip-compression-in-wordpress/#Enable_Gzip_Compression_on_Nginx_server)
  + [Results after Applying Compression](https://technumero.com/gzip-compression-in-wordpress/#Results_after_Applying_Compression)
  + [Enable Gzip Compression in WordPress using plugin](https://technumero.com/gzip-compression-in-wordpress/#Enable_Gzip_Compression_in_WordPress_using_plugin)
  + [Gzip Compression with WordPress Plugin – W3 Total Cache](https://technumero.com/gzip-compression-in-wordpress/#Gzip_Compression_with_WordPress_Plugin_W3_Total_Cache)
  + [Enable Gzip Compression with WordPress WP Super Cache Plugin](https://technumero.com/gzip-compression-in-wordpress/#Enable_Gzip_Compression_with_WordPress_WP_Super_Cache_Plugin)
  + [Use WP Fastest Cache Plugin to Enable Gzip Compression in WP](https://technumero.com/gzip-compression-in-wordpress/#Use_WP_Fastest_Cache_Plugin_to_Enable_Gzip_Compression_in_WP)
* [Wrapping it Up](https://technumero.com/gzip-compression-in-wordpress/#Wrapping_it_Up)
  + [General FAQ](https://technumero.com/gzip-compression-in-wordpress/#General_FAQ)
  + [What is Gzip Compression?](https://technumero.com/gzip-compression-in-wordpress/#What_is_Gzip_Compression-2)
  + [How do I enable Gzip Compression?](https://technumero.com/gzip-compression-in-wordpress/#How_do_I_enable_Gzip_Compression)
  + [Is Gzip enabled on my website?](https://technumero.com/gzip-compression-in-wordpress/#Is_Gzip_enabled_on_my_website)
  + [How do I enable Gzip Compression in WordPress?](https://technumero.com/gzip-compression-in-wordpress/#How_do_I_enable_Gzip_Compression_in_WordPress)

**What is Gzip Compression?**

Gzip is a file compressor that is used for compression and decompression of the files. It gives **best results compressing text files** (like stylesheets, HTML and JavaScript files).

When enabled on a website, Gzip compresses the files before transferring them from the server. Hence, it improves **pagespeed**, **loading time** and also **saves data usage** of the user. If you’re interested to know more about Gzip Compression check out these links [[1](https://en.wikipedia.org/wiki/Gzip), [2](https://www.youtube.com/watch?v=Mjab_aZsdxw)].

**How does it work?**

The concept of Gzip compression is quite easy.

It uses an algorithm which organizes repeated strings in a single place only, instead of saving those same strings again and again. And it manages those strings with their location values while compression and retrieving data back from compressed files.

Gzip compression works very well with **stylesheets** and **webpages** because all these resource files have many repeated strings.

Because of its efficient compression technique, Gzip may reduce the file size by 70-90%.

Let’s have a look at the given below example to understand the Gzip Compression…

Suppose if your HTML file contents following strings…

<h3>Enable Gzip Compression in WordPress </h3>

Then, the compressed file will look like this.

Hkg6dkGhJkjsdHkgHjL

But if your HTML file contents repetitive strings as following…

<h3>Enable Gzip Compression in WordPress</h3>

<h3>Enable Gzip Compression in WordPress via .htaccess</h3>

Then, Gzip compresses the repeated strings once and uses the latter part of your HTML file.

Hkg6dkGhJkjsdHkgHjL/GjDVjs

If you are working on *pagespeed* of your WordPress site then you should apply these too:

* [Remove Query Strings from Static Resources to Increase your Website Speed](https://technumero.com/remove-query-strings-from-static-resources/)
* [Minify JavaScript and CSS in WordPress to Increase PageSpeed](https://technumero.com/minify-javascript-and-css/)

**DEFLATE Compression vs Gzip Compression: The Difference**

You might have read on the web about Gzip Compression and DEFLATE compression.

If you haven’t, let me tell you about them in nutshell. Both compression algorithms (Gzip and DEFLATE) are different and used for different servers.

**Gzip compression** is used on **Apache** and **Nginx** servers, while **DEFLATE** is **only** used on **Apache** servers.

Before you start, you must know Gzip compression status of your website. Let’s see, how to do that.

**How to Check Gzip Compression**

First, you should check whether Gzip Compression is enabled or not on your WordPress site. You can use this [Gzip Test Tool](https://technumero.com/check-gzip-compression/) for accurate results or you can Google Gzip compression test tool.

[](https://technumero.com/check-gzip-compression/)Check if GZIP Compression is enabled or not?

If compression is already applied to your site. Then you may check if compression is enabled properly. Or *is there still scope for better compression?* (the technique which reduces the files-size most is better one)*.*

You also need to find out which web server software (Apache or Nginx) your web host is using. Because **Gzip Compression method** is different for both servers.

You also need to figure out, how to edit *.htaccess/Config* file. Here is a detailed guide [How to edit WordPress .htaccess file easily](https://technumero.com/create-and-edit-wordpress-htaccess-file/) you may refer.

**Enabling Gzip & DEFLATE Compression in WordPress**

You can enable Gzip compression in WordPress with or without using plugin.

1. **Enable Gzip Compression WordPress via .htaccess – without plugin method**
2. **Enabling Gzip Compression in WordPress using plugin**

**Enable Gzip Compression in WordPress via .htaccess**

As I have mentioned above, enabling Gzip Compression via htaccess/configuration file depends on the server i.e. Apache and Nginx, etc. So we’ll take both cases one by one. However, the process of editing the .htaccess (or configuration file in case of Nginx server) remains the same.

Step by step instructions to enable Gzip Compression in WordPress via .htaccess:

1. Log in to **Dashboard**/**cPanel** of your website.
2. Open **.htaccess** file (or config file in Nginx).
3. Paste the corresponding **code** (given below) based on server type i.e. Apache or Nginx.
4. **Save changes** and you’re done.
5. Now check your site on compression testing tools again. It will definitely reduce your website page size significantly and will increase page-load speed.

**Compression on Apache server**

Apache server supports **Gzip compression** as well as **DEFLATE compression**.

**Enable Gzip Compression on Apache server**

If you’re your website is hosted on Apache server, add following code in .htaccess file of your website.

# BEGIN GZIP COMPRESSION

<IfModule mod\_gzip.c>

mod\_gzip\_on Yes

mod\_gzip\_dechunk Yes

mod\_gzip\_item\_include file \.(html?|txt|css|js|php|pl)$

mod\_gzip\_item\_include handler ^cgi-script$

mod\_gzip\_item\_include mime ^text/.\*

mod\_gzip\_item\_include mime ^application/x-javascript.\*

mod\_gzip\_item\_exclude mime ^image/.\*

mod\_gzip\_item\_exclude rspheader ^Content-Encoding:.\*gzip.\*

</IfModule>

# END GZIP COMPRESSION

**Enable DEFLATE Compression in WordPress**

If you’re your website is hosted on **Apache server**, copy and paste the following code in WordPress .htaccess file of your website.

# BEGIN DEFLATE COMPRESSION

<IfModule mod\_filter.c>

AddOutputFilterByType DEFLATE "application/atom+xml" \

"application/javascript" \

"application/json" \

"application/ld+json" \

"application/manifest+json" \

"application/rdf+xml" \

"application/rss+xml" \

"application/schema+json" \

"application/vnd.geo+json" \

"application/vnd.ms-fontobject" \

"application/x-font-ttf" \

"application/x-javascript" \

"application/x-web-app-manifest+json" \

"application/xhtml+xml" \

"application/xml" \

"font/eot" \

"font/opentype" \

"image/bmp" \

"image/svg+xml" \

"image/vnd.microsoft.icon" \

"image/x-icon" \

"text/cache-manifest" \

"text/css" \

"text/html" \

"text/javascript" \

"text/plain" \

"text/vcard" \

"text/vnd.rim.location.xloc" \

"text/vtt" \

"text/x-component" \

"text/x-cross-domain-policy" \

"text/xml"

</IfModule>

# END DEFLATE COMPRESSION

**Compression on Nginx server**

Nginx server supports only Gzip compression. Let’s see to do that.

**Enable Gzip Compression on Nginx server**

If your website is hosted on Nginx server, paste the following code in the configuration file of your website.

gzip on;

gzip\_comp\_level 2;

gzip\_http\_version 1.0;

gzip\_proxied any;

gzip\_min\_length 1100;

gzip\_buffers 16 8k;

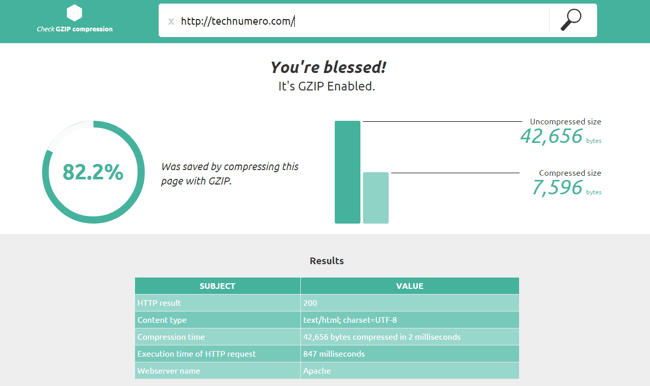
gzip\_types text/plain text/html text/css application/x-javascript text/xml application/xml application/xml+rss text/javascript;

gzip\_disable "MSIE [1-6].(?!.\*SV1)";

gzip\_vary on;

**Results after Applying Compression**

My blog TechNumero was hosted on Apache server, so I was using Gzip compression and DEFLATE compression both on my website. And it is reducing webpage size by 82.2%. Initially, my web page size is 42,656 Bytes (without compression) and after compression, my web page size is 7,596 Bytes only. Hence, I managed to save 35,060 Bytes using Gzip Compression on Technumero.com.

Gzip Compression in WordPress | Test Technumero.com

**Enable Gzip Compression in WordPress using plugin**

There are several plugins to the job at hand. Let’s discuss the most reliable and efficient ones.

**Gzip Compression with WordPress Plugin – W3 Total Cache**

W3 Total Cache is one of the most used caching plugins on WordPress platform. And if you are using this plugin, you can enable Gzip compression via this plugin.

Step by step instructions to enable Gzip compression using W3 Total Cache WordPress plugin:

* Go to WordPress **Dashboard** > W3 Total Cache **Plugin Settings** page*.*
* Navigate to **Browser Cache page***.*
* And *Check* the **Enable HTTP Compression**
* Don’t forget to *click on* **Save changes** *button* and you are done.

Enable Gzip Compression using WordPress Plugin – W3 Total Cache

**Enable Gzip Compression with WordPress WP Super Cache Plugin**

WP Super Cache is another popular caching plugin. And you can enable Gzip compression with just one click using this plugin.

Follow steps the given below to Enable Gzip compression with WP Super Cache plugin:

* Navigate to WP Dashboard > WP Super Cache **Plugin Settings**
* *Click* on the **Advanced Tab** and
* Then **Enable the First option** in **Miscellaneous settings.**
* **Save** the *changes* and you’re done.

Enable Gzip Compression with WordPress Plugin – WP Super Cache

**Use WP Fastest Cache Plugin to Enable Gzip Compression in WP**

WP Fastest Cache Plugin is another plugin which you can to enable Gzip compression in WordPress easily.

Followings are the easy steps to Enable Gzip compression using WP Fastest Cache plugin:

* Go to WordPress **Dashboard** > **WP Fastest Cache** > **Settings**.
* Under **Settings** Tab *check* the against **Gzip** to enable it.
* Click Submit to save *changes* and you’re done.

Use WP Fastest Cache Plugin to Enable Gzip Compression

**Wrapping it Up**

Gzip Compression is one of the most used technique for file compression. In addition to that, it is also one of the most recommended methods by pro bloggers for PageSpeed Optimization.

Nowadays, Gzip Compressions comes enabled by default with few modern web hosting servers. However, if it is not enabled on your server, now you know, how to enable Gzip Compression.

Feel free to fire your queries via the comment section below. Do let us know, how much file size you have reduced by enabling Gzip Compression?

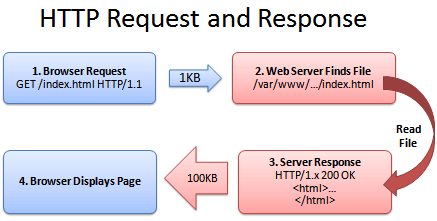
# How To Optimize Your Site With GZIP Compression

Compression is a simple, effective way to save bandwidth and speed up your site. I hesitated when recommending gzip compression when [speeding up your javascript](https://betterexplained.com/articles/speed-up-your-javascript-load-time/) because of [problems](https://support.microsoft.com/kb/837251) [in](https://support.microsoft.com/kb/823386) [older browsers](http://schroepl.net/projekte/mod_gzip/browser.htm).

But it’s the 21st century. Most of my traffic comes from modern browsers, and quite frankly, most of [my users](http://instacalc.com) are fairly tech-savvy. I don’t want to slow everyone else down because somebody is chugging along on IE 4.0 on Windows 95. Google and Yahoo use gzip compression. A modern browser is needed to enjoy modern web content and modern web speed — so gzip encoding it is. Here’s how to set it up.

## Wait, wait, wait: Why are we doing this?

Before we start I should explain what content encoding is. When you request a file like http://www.yahoo.com/index.html, your browser talks to a web server. The conversation goes a little like this:



1. Browser: Hey, **GET** me /index.html
2. Server: Ok, let me see if index.html is lying around…
3. Server: Found it! Here’s your response code (200 OK) and I’m sending the file.
4. Browser: 100KB? Ouch… waiting, waiting… ok, it’s loaded.

Of course, the actual headers and protocols are much more formal (monitor them with [Live HTTP Headers](https://betterexplained.com/articles/how-to-debug-web-applications-with-firefox/) if you’re so inclined).

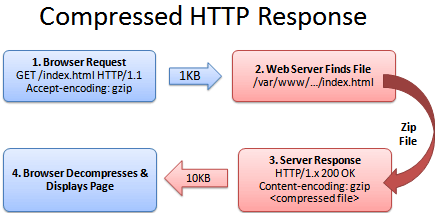
But it worked, and you got your file.

## So what’s the problem?

Well, the system works, but it’s not that efficient. 100KB is a **lot of text**, and frankly, HTML is redundant. Every <html>, <table> and <div> tag has a closing tag that’s almost the same. Words are repeated throughout the document. Any way you slice it, HTML (and its beefy cousin, XML) is not lean.

And what’s the plan when a file’s too big? Zip it!

If we could send a .zip file to the browser (index.html.zip) instead of plain old index.html, we’d save on bandwidth and download time. The browser could download the zipped file, extract it, and then show it to user, who’s in a good mood because the page loaded quickly. The browser-server conversation might look like this:



1. Browser: Hey, can I **GET** index.html? I’ll take a compressed version if you’ve got it.
2. Server: Let me find the file… yep, it’s here. And you’ll take a compressed version? Awesome.
3. Server: Ok, I’ve found index.html (200 OK), am zipping it and sending it over.
4. Browser: Great! It’s only 10KB. I’ll unzip it and show the user.

The formula is simple: Smaller file = faster download = **happy user**.

Don’t believe me? The HTML portion of the yahoo home page goes from 101kb to 15kb after compression:

yahoo_compression.PNG

## The (not so) hairy details

The tricky part of this exchange is the browser and server knowing it’s ok to send a zipped file over. The agreement has two parts

* The **browser sends a header** telling the server it accepts compressed content (gzip and deflate are two compression schemes): Accept-Encoding: gzip, deflate
* The **server sends a response** if the content is actually compressed: Content-Encoding: gzip

If the server doesn’t send the content-encoding response header, it means the file is not compressed (the default on many servers). The “Accept-encoding” header is just a request by the browser, not a demand. If the server doesn’t want to send back compressed content, the browser has to make do with the heavy regular version.

## Setting up the server

The “good news” is that we can’t control the browser. It either sends the Accept-encoding: gzip, deflate header or it doesn’t.

Our job is to configure the server so it returns zipped content if the browser can handle it, saving bandwidth for everyone (and giving us a happy user).

For IIS, [enable compression](https://technet.microsoft.com/en-us/library/cc771003(WS.10).aspx) in the settings.

In Apache, [enabling output compression](https://httpd.apache.org/docs/2.0/mod/mod_deflate.html) is fairly straightforward. Add the following to your .htaccess file:

# compress text, html, javascript, css, xml:

AddOutputFilterByType DEFLATE text/plain

AddOutputFilterByType DEFLATE text/html

AddOutputFilterByType DEFLATE text/xml

AddOutputFilterByType DEFLATE text/css

AddOutputFilterByType DEFLATE application/xml

AddOutputFilterByType DEFLATE application/xhtml+xml

AddOutputFilterByType DEFLATE application/rss+xml

AddOutputFilterByType DEFLATE application/javascript

AddOutputFilterByType DEFLATE application/x-javascript

# Or, compress certain file types by extension:

<files \*.html>

SetOutputFilter DEFLATE

</files>

Apache actually has two compression options:

* **mod\_deflate** is easier to set up and is standard.
* **mod\_gzip** seems more powerful: you can pre-compress content.

Deflate is quick and works, so I use it; use mod\_gzip if that floats your boat. In either case, Apache checks if the browser sent the “Accept-encoding” header and returns the compressed or regular version of the file. However, some older browsers may have trouble (more below) and there are special directives you can add to correct this.

If you can’t change your .htaccess file, you can [use PHP](https://perishablepress.com/press/2007/03/26/fast-effective-php-compression/) to return compressed content. Give your HTML file a .php extension and add this code to the top:

In PHP:

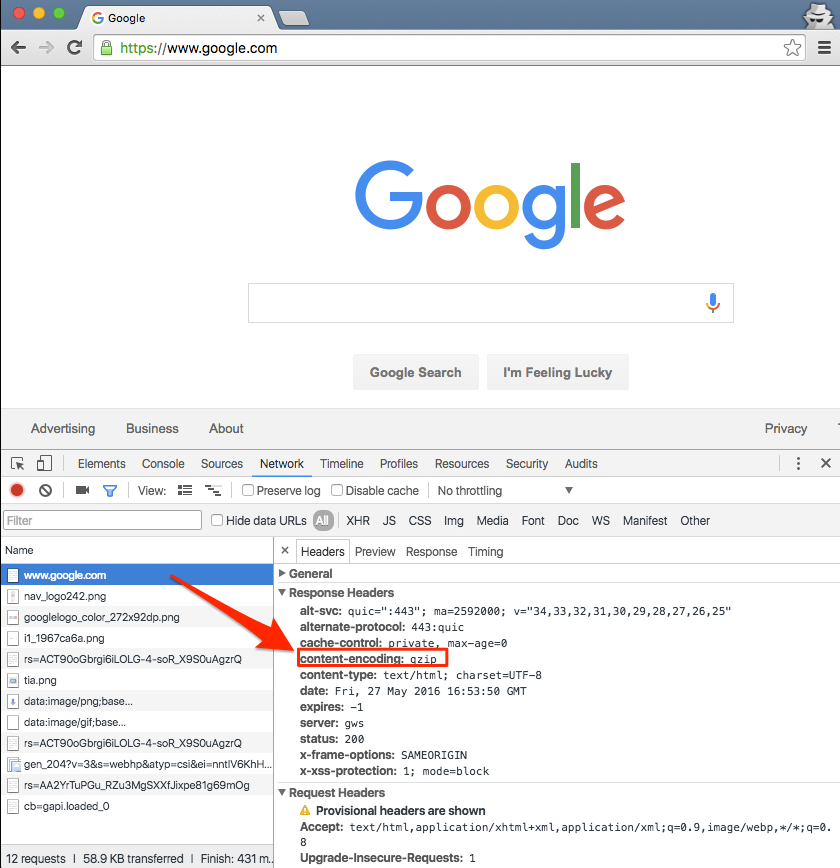
<?php if (substr\_count($\_SERVER[‘HTTP\_ACCEPT\_ENCODING’], ‘gzip’)) ob\_start(“ob\_gzhandler”); else ob\_start(); ?>

We check the “Accept-encoding” header and return a gzipped version of the file (otherwise the regular version). This is almost like building your own webserver (what fun!). But really, try to use Apache to compress your output if you can help it. You don’t want to monkey with your files.

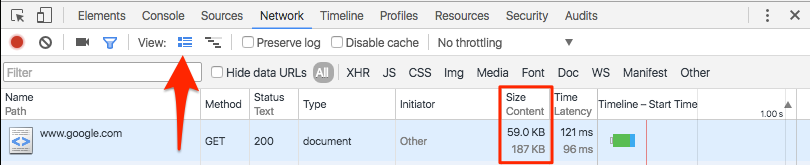
## Verify Your Compression

Once you’ve configured your server, check to make sure you’re actually serving up compressed content.

* **Online:** Use the [online gzip test](http://www.gidnetwork.com/tools/gzip-test.php) to check whether your page is compressed.
* **In your browser:** In Chrome, open the Developer Tools > Network Tab (Firefox/IE will be similar). Refresh your page, and click the network line for the page itself (i.e., www.google.com). The header “Content-encoding: gzip” means the contents were sent compressed.



Click the “Use large rows” icon to get more details, including the compressed transfer size and the true content size.



Be prepared to marvel at the results. The [instacalc homepage](http://instacalc.com) shrunk from 36k to 10k, a 75% reduction in size.

## Try Some Examples

I’ve set up some pages and a [downloadable example](https://betterexplained.com/examples/compressed/compression-example.zip):

* [index.html](https://betterexplained.com/examples/compressed/index.html) – No explicit compression (on this server, I am using compression by default).
* [index.htm](https://betterexplained.com/examples/compressed/index.htm) – Explicitly compressed with Apache .htaccess using \*.htm as a rule
* [index.php](https://betterexplained.com/examples/compressed/index.php) – Explicitly compressed using the PHP header

Feel free to download the files, put them on your server and tweak the settings.

## Caveats

As exciting as it may appear, HTTP Compression isn’t all fun and games. Here’s what to watch out for:

* **Older browsers**: Yes, some browsers still may have trouble with compressed content (they say they can accept it, but really they can’t). If your site absolutely must work with Netscape 1.0 on Windows 95, you may not want to use HTTP Compression. Apache mod\_deflate [has some rules](https://httpd.apache.org/docs/2.0/mod/mod_deflate.html#recommended) to avoid compression for older browsers.
* **Already-compressed content**: Most images, music and videos are already compressed. Don’t waste time compressing them again. In fact, you probably only need to compress the “big 3” (HTML, CSS and Javascript).
* **CPU-load**: Compressing content on-the-fly uses CPU time and saves bandwidth. Usually this is a great tradeoff given the speed of compression. There are ways to pre-compress static content and send over the compressed versions. This requires more configuration; even if it’s not possible, compressing output may still be a net win. Using CPU cycles for a faster user experience is well worth it, given the short attention spans on the web.

Enabling compression is one of the fastest ways to improve your site’s performance. Go forth, set it up, and let your users enjoy the benefits.