

```

# Apache Server Configs v2.8.0 | MIT License
# https://github.com/h5bp/server-configs-apache

# (!) Using `.htaccess` files slows down Apache, therefore, if you have access
# to the main server config file (usually called `httpd.conf`), you should add
# this logic there: http://httpd.apache.org/docs/current/howto/htaccess.html.

# #####
# # CROSS-ORIGIN RESOURCE SHARING (CORS) #
# #####

# -----
# | Cross-origin requests |
# -----

# Allow cross-origin requests.

# http://enable-cors.org/
# http://www.w3.org/TR/cors/
# https://code.google.com/p/html5security/wiki/CrossOriginRequestSecurity

# <IfModule mod_headers.c>
#     Header set Access-Control-Allow-Origin "*"
# </IfModule>

# -----

# By default allow cross-origin access to web fonts.

<IfModule mod_headers.c>
    <FilesMatch "\.(eot|otf|tt[cf]|woff2?)$">
        Header set Access-Control-Allow-Origin "*"
    </FilesMatch>
</IfModule>

# -----
# | Cross-origin resource timing |
# -----

# Allow cross-origin access to the timing information for all resources.

```

```

# If a resource isn't served with a `Timing-Allow-Origin` header that would
# allow its timing information to be shared with the current document, some of
# the attributes of the `PerformanceResourceTiming` object will be set to zero.

# http://www.w3.org/TR/resource-timing/

# <IfModule mod_headers.c>
#     Header set Timing-Allow-Origin: "*"
# </IfModule>

# -----
# | CORS-enabled images |
# -----

# Send the CORS header for images when browsers request it.

# https://developer.mozilla.org/en-US/docs/Web/HTML/CORS_enabled_image
# http://blog.chromium.org/2011/07/using-cross-domain-images-in-webgl-and.html
# http://hacks.mozilla.org/2011/11/using-cors-to-load-webgl-textures-from-cross-domain-images/

<IfModule mod_setenvif.c>
    <IfModule mod_headers.c>
        <FilesMatch "\.(curl|gif|ico|jpe?g|png|svgz?|webp)$">
            SetEnvIf Origin ":" IS_CORS
            Header set Access-Control-Allow-Origin "*" env=IS_CORS
        </FilesMatch>
    </IfModule>
</IfModule>

# #####
# # ERRORS #
# #####

# -----
# | 404 error prevention |
# -----

# Disable the pattern matching based on filenames.

```

```
# This setting prevents Apache from returning a 404 error as the result
# of a rewrite when the directory with the same name does not exist.
```

```
# http://httpd.apache.org/docs/current/content-negotiation.html#multiviews
# http://www.webmasterworld.com/apache/3808792.htm
```

## Options -MultiViews

```
# -----
# | Custom error messages / pages |
# -----
```

```
# Customize what Apache returns to the client in case of an error.
# http://httpd.apache.org/docs/current/mod/core.html#errordocument
```

## ErrorDocument 404 /404.html

```
# #####
# # INTERNET EXPLORER #
# #####
```

```
# -----
# | Better website experience |
# -----
```

```
# Force Internet Explorer to render pages in the highest available
# mode in the various cases when it may not.
# https://hsivonen.fi/doctype/#ie8
```

```
<IfModule mod_headers.c>
```

```
Header set X-UA-Compatible "IE=edge"
```

```
# `mod_headers` cannot match based on the content-type, however, this header
# should be send only for HTML documents and not for the other resources
```

```
<FilesMatch "\.(appcache|atom|crx|css|curl|eot|f4[abpv]|flv|geojson|gif|htc|ico|jpe?
gl|jsl|json(1d)?
|m4[av]|manifest|map|mp4|oex|og[agv]|opus|otf|pdf|png|rdf|rss|safariextz|svgz?
|swf|topojson|tt[cf]|txt|vcf|vtt|webapp|web[mp]|woff2?|xml|xpi)$">
```

```
Header unset X-UA-Compatible
```

```
</FilesMatch>
```

```
</IfModule>
```

```

# -----
# | Cookie setting from iframes |
# -----

# Allow cookies to be set from iframes in Internet Explorer.

# http://msdn.microsoft.com/en-us/library/ms537343.aspx
# http://www.w3.org/TR/2000/CR-P3P-20001215/

# <IfModule mod_headers.c>
#     Header set P3P "policyref=\"/w3c/p3p.xml\", CP=\"IDC DSP COR ADM DEVI TAIi PSA PSD
IVAI IVDi CONi HIS OUR IND CNT\""
# </IfModule>

# #####
# # MEDIA TYPES AND CHARACTER ENCODINGS #
# #####

# -----
# | Media types |
# -----

# Serve resources with the proper media types (formerly known as MIME types).
# http://www.iana.org/assignments/media-types/media-types.xhtml

<IfModule mod_mime.c>

# Audio
AddType audio/mp4 f4a f4b m4a
AddType audio/ogg oga ogg opus

# Data interchange
AddType application/json json map topojson
AddType application/ld+json jsonld
AddType application/vnd.geo+json geojson

# JavaScript
# Normalize to standard type.
# http://tools.ietf.org/html/rfc4329#section-7.2
AddType application/javascript js

```

## # Manifest files

```
# If you are providing a web application manifest file (see the
# specification: http://w3c.github.io/manifest/), it is recommended
# that you serve it with the `application/manifest+json` media type.
#
# Because the web application manifest file doesn't have its own
# unique file extension, you can set its media type either by matching:
#
# 1) the exact location of the file (this can be done using a directive
#    such as ``, but it will NOT work in the `.htaccess` file,
#    so you will have to do it in the main server configuration file or
#    inside of a `` container)
```

```
# e.g.:
```

```
#     <Location "/.well-known/manifest.json">
#         AddType application/manifest+json          json
#     </Location>
```

```
# 2) the filename (this can be problematic as you will need to ensure
#    that you don't have any other file with the same name as the one
#    you gave to your web application manifest file)
```

```
# e.g.:
```

```
#     <Files "manifest.json">
#         AddType application/manifest+json          json
#     </Files>
```

```
AddType application/x-web-app-manifest+json    webapp
AddType text/cache-manifest                      appcache manifest
```

## # Video

```
AddType video/mp4                               f4v f4p m4v mp4
AddType video/ogg                               ogv
AddType video/webm                              webm
AddType video/x-flv                             flv
```

## # Web fonts

```
AddType application/font-woff                  woff
```

```
AddType application/font-woff2          woff2
AddType application/vnd.ms-fontobject    eot
```

```
# Browsers usually ignore the font media types and simply sniff
# the bytes to figure out the font type.
# http://mimesniff.spec.whatwg.org/#matching-a-font-type-pattern
```

```
# Chrome however, shows a warning if any other media types are used
# for the following two font types.
```

```
AddType application/x-font-ttf          ttc ttf
AddType font/opentype                    otf
```

```
AddType image/svg+xml                  svg svgz
```

```
# Other
```

```
AddType application/octet-stream        safariextz
AddType application/x-chrome-extension    crx
AddType application/x-opera-extension     oex
AddType application/x-xpinstall           xpi
AddType application/xml                   atom rdf rss xml
AddType image/webp                        webp
AddType image/x-icon                      cur ico
AddType text/vtt                          vtt
AddType text/x-component                  htc
AddType text/x-vcard                      vcf
```

```
</IfModule>
```

```
# -----
# | Character encodings |
# -----
```

```
# Set `UTF-8` as the character encoding for all resources served with
# the media type of `text/html` or `text/plain`.
```

```
AddDefaultCharset utf-8
```

```
# Set `UTF-8` as the character encoding for other certain resources.
```

```
<IfModule mod_mime.c>
```

```
    AddCharset utf-8 .atom \
                .css \
```

```
.geojson \  
.js \  
.json \  
.jsonld \  
.rdf \  
.rss \  
.topojson \  
.vtt \  
.webapp \  
.xml
```

```
</IfModule>
```

```
# #####  
# # URL REWRITES #  
# #####
```

```
# -----  
# | Rewrite engine |  
# -----
```

```
# (1) Turn on the rewrite engine  
# (this is necessary in order for the `RewriteRule` directives to work).  
# http://httpd.apache.org/docs/current/mod/mod\_rewrite.html#RewriteEngine  
#  
# (2) Enable the `FollowSymLinks` option if it isn't already.  
# http://httpd.apache.org/docs/current/mod/core.html#options  
#  
# (3) If your web host doesn't allow the `FollowSymLinks` option, you may  
# need to comment it out and use `Options +SymLinksIfOwnerMatch`, but  
# be aware of the performance impact.  
# http://httpd.apache.org/docs/current/misc/perf-tuning.html#symlinks  
#  
# (4) Some cloud hosting services will also require `RewriteBase` to be set.  
# http://www.rackspace.com/knowledge\_center/frequently-asked-question/why-is-modrewrite-not-working-on-my-site  
#  
# (5) Depending on how your server is set up, you may need to use the  
# `RewriteOptions` directive to enable some options for the rewrite engine.  
# http://httpd.apache.org/docs/current/mod/mod\_rewrite.html#rewriteoptions
```

```
<IfModule mod_rewrite.c>
```

```
# (1)
RewriteEngine On

# (2)
Options +FollowSymlinks

# (3)
# Options +SymLinksIfOwnerMatch

# (4)
# RewriteBase /

# (5)
# RewriteOptions <options>

</IfModule>

# -----
# | Suppressing / Forcing the `www.` at the beginning of URLs |
# -----

# The same content should never be available under two different URLs,
# especially not with and without `www.` at the beginning. This can cause
# SEO problems (duplicate content), and therefore, you should choose one
# of the alternatives and redirect the other one.

# By default `Option 1` (no `www.`) is activated.
# http://no-www.org/faq.php?q=class_b

# If you would prefer to use `Option 2`, just comment out all the lines
# from `Option 1` and uncomment the ones from `Option 2`.

# IMPORTANT: NEVER USE BOTH RULES AT THE SAME TIME!

# -----

# Option 1: rewrite www.example.com → example.com

<IfModule mod_rewrite.c>
    RewriteCond %{HTTPS} !=on
```



```

-----
RewriteCond %{HTTP_HOST} ^www\.(.+$) [NC]
RewriteRule ^ http://%1%{REQUEST_URI} [R=301,L]
</IfModule>

# -----

# Option 2: rewrite example.com + www.example.com

# Be aware that the following might not be a good idea if you use "real"
# subdomains for certain parts of your website.

# <IfModule mod_rewrite.c>
#     RewriteCond %{HTTPS} !=on
#     RewriteCond %{HTTP_HOST} !^www\. [NC]
#     RewriteCond %{SERVER_ADDR} !=127.0.0.1
#     RewriteCond %{SERVER_ADDR} !=::1
#     RewriteRule ^ http://www.%{HTTP_HOST}%{REQUEST_URI} [R=301,L]
# </IfModule>

# #####
# # SECURITY #
# #####

# -----
# | Clickjacking |
# -----

# Protect website against clickjacking.

# The example below sends the `X-Frame-Options` response header with the value
# `DENY`, informing browsers not to display the web page content in any frame.

# This might not be the best setting for everyone. You should read about the
# other two possible values for `X-Frame-Options`: `SAMEORIGIN` & `ALLOW-FROM`.
# http://tools.ietf.org/html/rfc7034#section-2.1

# Keep in mind that while you could send the `X-Frame-Options` header for all
# of your site's pages, this has the potential downside that it forbids even
# non-malicious framing of your content (e.g.: when users visit your site using
# a Google Image Search results page).

```

```
# Nonetheless, you should ensure that you send the `X-Frame-Options` header for
# all pages that allow a user to make a state changing operation (e.g. pages
# that contain one-click purchase links, checkout or bank-transfer confirmation
# pages, pages that make permanent configuration changes, etc.).

# Sending the `X-Frame-Options` header can also protect your website against
# more than just clickjacking attacks: https://cure53.de/xfo-clickjacking.pdf.

# http://tools.ietf.org/html/rfc7034
# http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-
frame-options.aspx
# https://www.owasp.org/index.php/Clickjacking

# <IfModule mod_headers.c>
#     Header set X-Frame-Options "DENY"
#     <FilesMatch "\.(appcache|atom|crx|css|curl|eot|f4[abpw]|flv|geo.json|gif|htc|icol|jpe?
gl|js|json|1d)?
|m4[av]|manifest|map|mp4|oex|og[agv]|opus|otf|pdf|png|rdf|rss|safariextz|svgz?
|swf|topo.json|tt[cf]|txt|vcf|vtt|webapp|web[mp]|woff2?|xml|xpi)$">
#         Header unset X-Frame-Options
#     </FilesMatch>
# </IfModule>

# -----
# | Content Security Policy (CSP) |
# -----

# Mitigate the risk of cross-site scripting and other content-injection attacks.

# This can be done by setting a `Content Security Policy` which whitelists
# trusted sources of content for your website.

# The example header below allows ONLY scripts that are loaded from the current
# site's origin (no inline scripts, no CDN, etc). This almost certainly won't
# work as-is for your site!

# For more details on how to craft a reasonable policy for your site, read:
# http://www.html5rocks.com/en/tutorials/security/content-security-policy/ (or
# the specification: http://www.w3.org/TR/CSP11/). Also, to make things easier,
# you can use an online CSP header generator such as: http://cspisawesome.com/.
```

```
# <IfModule mod_headers.c>
#   Header set Content-Security-Policy "script-src 'self'; object-src 'self'"
#   <FilesMatch "\.(appcache|atom|crx|css|curl|eot|f4[abpw]|flv|geojson|gif|html|icol|jpe?
gl|js|json|ld)?
|m4[av]|manifest|map|mp4|oex|og[agv]|opus|otf|pdf|png|rdf|rss|safariextz|svgz?
|swf|topojson|ttf|cf|txt|vcf|vtt|webapp|web[mp]|woff2?|xml|xpi)$">
#       Header unset Content-Security-Policy
#   </FilesMatch>
# </IfModule>
```

```
# -----
# | File access |
# -----
```

*# Block access to directories without a default document.*

*# You should leave the following uncommented, as you shouldn't allow anyone to  
# surf through every directory on your server (which may includes rather private  
# places such as the CMS's directories).*

```
<IfModule mod_autoindex.c>
    Options -Indexes
</IfModule>
```

```
# -----
```

*# Block access to all hidden files and directories with the exception of the  
# visible content from within the `/.well-known/` hidden directory.*

*# These types of files usually contain user preferences or the preserved state  
# of an utility, and can include rather private places like, for example, the  
# `.git` or `.svn` directories.*

*# The `/.well-known/` directory represents the standard (RFC 5785) path prefix  
# for "well-known locations" (e.g.: `/.well-known/manifest.json`,  
# `/.well-known/keybase.txt`), and therefore, access to its visible content  
# should not be blocked.*

```
# https://www.mnot.net/blog/2010/04/07/well-known
# http://tools.ietf.org/html/rfc5785
```

```

<IfModule mod_rewrite.c>
    RewriteCond %{REQUEST_URI} "!^(^/)\.well-known/([^. /]+. /?)+$" [NC]
    RewriteCond %{SCRIPT_FILENAME} -d [OR]
    RewriteCond %{SCRIPT_FILENAME} -f
    RewriteRule "(^/)\." - [F]
</IfModule>

# -----

# Block access to files that can expose sensitive information.

# By default, block access to backup and source files that may be left by some
# text editors and can pose a security risk when anyone has access to them.
# http://feross.org/cmsexploit/

# IMPORTANT: Update the `<FilesMatch>` regular expression from below to include
# any files that might end up on your production server and can expose sensitive
# information about your website. These files may include: configuration files,
# files that contain metadata about the project (e.g.: project dependencies),
# build scripts, etc..

<FilesMatch "(^#. *#| \.(bak|conf|dist|fla|in[ci]|log|psd|sh|sql|sw[op])|")$" >

    # Apache < 2.3
    <IfModule !mod_authz_core.c>
        Order allow,deny
        Deny from all
        Satisfy All
    </IfModule>

    # Apache ≥ 2.3
    <IfModule mod_authz_core.c>
        Require all denied
    </IfModule>

</FilesMatch>

# -----
# | Reducing MIME type security risks |
# -----

```

```
# Prevent some browsers from MIME-sniffing the response.

# This reduces exposure to drive-by download attacks and cross-origin data
# leaks, and should be left uncommented, especially if the web server is
# serving user-uploaded content or content that could potentially be treated
# as executable by the browser.

# http://www.slideshare.net/hasegawayosuke/owasp-hasegawa
# http://blogs.msdn.com/b/ie/archive/2008/07/02/ie8-security-part-v-comprehensive-
# protection.aspx
# http://msdn.microsoft.com/en-us/library/ie/gg622941.aspx
# http://mimesniff.spec.whatwg.org/

<IfModule mod_headers.c>
    Header set X-Content-Type-Options "nosniff"
</IfModule>

# -----
# | Reflected Cross-Site Scripting (XSS) attacks |
# -----

# (1) Try to re-enable the Cross-Site Scripting (XSS) filter built into the
#     most recent web browsers.
#
#     The filter is usually enabled by default, but in some cases it may be
#     disabled by the user. However, in Internet Explorer for example, it can
#     be re-enabled just by sending the `X-XSS-Protection` header with the
#     value of `1`.
#
# (2) Prevent web browsers from rendering the web page if a potential reflected
#     (a.k.a. a non-persistent) XSS attack is detected by the filter.
#
#     By default, if the filter is enabled and browsers detect a reflected
#     XSS attack, they will attempt to block the attack by making the smallest
#     possible modifications to the returned web page.
#
#     Unfortunately, in some browsers (e.g.: Internet Explorer), this default
#     behavior may allow the XSS filter to be exploited, thereby, it's better
#     to tell browsers to prevent the rendering of the page altogether, instead
#     of attempting to modify it.
```

```

#
# http://hackademix.net/2009/11/21/ies-xss-filter-creates-xss-vulnerabilities
#
# IMPORTANT: Do not rely on the XSS filter to prevent XSS attacks! Ensure that
# you are taking all possible measures to prevent XSS attacks, the most obvious
# being: validating and sanitizing your site's inputs.
#
# http://blogs.msdn.com/b/ie/archive/2008/07/02/ie8-security-part-iv-the-xss-filter.aspx
# http://blogs.msdn.com/b/ieinternals/archive/2011/01/31/controlling-the-internet-explorer-xss-filter-with-the-x-xss-protection-http-header.aspx
# https://www.owasp.org/index.php/Cross-site\_Scripting\_%28XSS%29

# <IfModule mod_headers.c>
#     #                                     (1)     (2)
#     Header set X-XSS-Protection "1; mode=block"
#     <FilesMatch "\.(appcache|atom|crx|css|curl|eot|f4[abpv]|flv|geojson|gif|htcl|icol|jpe?
gl|jsl|json|ld)?
|m4[av]|manifest|map|mp4|oex|og[agv]|opus|otf|pdf|png|rdf|rss|safari|extz|svgz?
|swf|topojson|tt[cf]|txt|vcf|vtt|webapp|web[mp]|woff2?|xml|xpi)$">
#         Header unset X-XSS-Protection
#     </FilesMatch>
# </IfModule>

# -----
# | Secure Sockets Layer (SSL) |
# -----

# Rewrite secure requests properly in order to prevent SSL certificate warnings.
# E. g.: prevent `https://www.example.com` when your certificate only allows
# `https://secure.example.com`.

# <IfModule mod_rewrite.c>
#     RewriteCond %{SERVER_PORT} !^443
#     RewriteRule ^ https://example-domain-please-change-me.com%{REQUEST_URI} [R=301,L]
# </IfModule>

# -----
# | HTTP Strict Transport Security (HSTS) |
# -----

# Force client-side SSL redirection.

```



```

<IfModule mod_deflate.c>

    # Force compression for mangled headers.
    # https://developer.yahoo.com/blogs/ymn/pushing-beyond-gzipping-25601.html

    <IfModule mod_setenvif.c>
        <IfModule mod_headers.c>
            SetEnvIfNoCase ^(\Accept-EncodXngl X-cept-Encodingl X(15)|^(15)|-(15))$
^((gzip|deflate)\s*, ?\s*)+([X"-]{4,13})$ HAVE_Accept-Encoding
            RequestHeader append Accept-Encoding "gzip, deflate" env=HAVE_Accept-Encoding
        </IfModule>
    </IfModule>

# -----

# Map certain file types to the specified encoding type in order to
# make Apache serve them with the appropriate `Content-Encoding` HTTP
# response header (this will NOT make Apache compress them!).

# If the following file types wouldn't be served without the appropriate
# `Content-Enable` HTTP response header, client applications (e.g.:
# browsers) wouldn't know that they first need to uncompress the response,
# and thus, wouldn't be able to understand the content.

# http://httpd.apache.org/docs/current/mod/mod_mime.html#addencoding

<IfModule mod_mime.c>
    AddEncoding gzip          svgz
</IfModule>

# -----

# Compress all output labeled with one of the following media types.

# IMPORTANT: For Apache versions below 2.3.7 you don't need to enable
# `mod_filter` and can remove the `<IfModule mod_filter.c>` & `</IfModule>`
# lines as `AddOutputFilterByType` is still in the core directives.

<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-web-app-manifest+json" \
"application/xhtml+xml" \
"application/xml" \
"font/opentype" \
"image/svg+xml" \
"image/x-icon" \
"text/cache-manifest" \
"text/css" \
"text/html" \
"text/javascript" \
"text/plain" \
"text/vtt" \
"text/x-component" \
"text/xml"
```

```
</IfModule>
```

```
</IfModule>
```

```
# -----
# | Content transformation |
# -----
```

```
# Prevent mobile network providers from modifying the website's content.
# http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html #sec14.9.5.
```

```
# <IfModule mod_headers.c>
#   Header merge Cache-Control "no-transform"
# </IfModule>
```

```
# -----
# | ETags |
# -----
```

```

..

# Remove `ETags` as resources are sent with far-future expires headers.
# https://developer.yahoo.com/performance/rules.html#etags

# `FileETag None` doesn't work in all cases.
<IfModule mod_headers.c>
    Header unset ETag
</IfModule>

FileETag None

# -----
# | Expires headers |
# -----

# Serve resources with far-future expires headers.

# IMPORTANT: If you don't control versioning with filename-based cache
# busting, consider lowering the cache times to something like one week.

<IfModule mod_expires.c>

    ExpiresActive on
    ExpiresDefault "access plus 1 month"

# CSS
    ExpiresByType text/css "access plus 1 year"

# Data interchange
    ExpiresByType application/json "access plus 0 seconds"
    ExpiresByType application/ld+json "access plus 0 seconds"
    ExpiresByType application/schema+json "access plus 0 seconds"
    ExpiresByType application/vnd.geo+json "access plus 0 seconds"
    ExpiresByType application/xml "access plus 0 seconds"
    ExpiresByType text/xml "access plus 0 seconds"

# Favicon (cannot be renamed!) and cursor images
    ExpiresByType image/x-icon "access plus 1 week"

# HTML components (HTCs)
    ExpiresByType text/x-component "access plus 1 month"

```

## # HTML

ExpiresByType text/html "access plus 0 seconds"

## # JavaScript

ExpiresByType application/javascript "access plus 1 year"

ExpiresByType text/javascript "access plus 1 year"

## # Manifest files

ExpiresByType application/manifest+json "access plus 1 year"

ExpiresByType application/x-web-app-manifest+json "access plus 0 seconds"

ExpiresByType text/cache-manifest "access plus 0 seconds"

## # Media

ExpiresByType audio/ogg "access plus 1 month"

ExpiresByType image/gif "access plus 1 month"

ExpiresByType image/jpeg "access plus 1 month"

ExpiresByType image/png "access plus 1 month"

ExpiresByType video/mp4 "access plus 1 month"

ExpiresByType video/ogg "access plus 1 month"

ExpiresByType video/webm "access plus 1 month"

## # Web feeds

ExpiresByType application/atom+xml "access plus 1 hour"

ExpiresByType application/rdf+xml "access plus 1 hour"

ExpiresByType application/rss+xml "access plus 1 hour"

## # Web fonts

ExpiresByType application/font-woff "access plus 1 month"

ExpiresByType application/font-woff2 "access plus 1 month"

ExpiresByType application/vnd.ms-fontobject "access plus 1 month"

ExpiresByType application/x-font-ttf "access plus 1 month"

ExpiresByType font/opentype "access plus 1 month"

ExpiresByType image/svg+xml "access plus 1 month"

</IfModule>

# -----  
# | *Filename-based cache busting* |  
# -----

```

# If you're not using a build process to manage your filename version revving,
# you might want to consider enabling the following directives to route all
# requests such as `/css/style.12345.css` to `/css/style.css`.

# To understand why this is important and a better idea than `*.css?v231`, read:
# http://www.stevesouders.com/blog/2008/08/23/revving-filenames-dont-use-querystring/

# <IfModule mod_rewrite.c>
#     RewriteCond %{REQUEST_FILENAME} !-f
#     RewriteRule ^(\.+)\/(\d+)\. (css|curl|gif|ico|jpe?g|js|png|svgz?|webp)$ $1. $3 [L]
# </IfModule>

# -----
# | File concatenation |
# -----

# Allow concatenation from within specific files.

# e.g.:
#
# If you have the following lines in a file called, for example,
# `main.combined.js`:
#
#     <!--#include file="js/jquery.js" -->
#     <!--#include file="js/jquery.timer.js" -->
#
# Apache will replace those lines with the content of the specified files.

# <IfModule mod_include.c>
#
#     <FilesMatch "\.combined\.js$" >
#         Options +Includes
#         AddOutputFilterByType INCLUDES application/javascript text/javascript
#         SetOutputFilter INCLUDES
#     </FilesMatch>
#
#     <FilesMatch "\.combined\.css$" >
#         Options +Includes
#         AddOutputFilterByType INCLUDES text/css
#         SetOutputFilter INCLUDES
#     </FilesMatch>

```

```
# </FilesMatch>
#
# </IfModule>
```