Mobile Site Speed Optimization Master Cookbook Guide

VERSION: 1.0

Table of Contents

Enabling Browser Caching for Apache

Browser caching is a technique to store some files in local browser. When a browser loads a webpage it downloads all the web files for properly displaying the page which includes all the images, HTML, CSS, and JavaScript. To enable browser caching, the web server's HTTP headers need to be edited. This needs to be followed by setting the expiration time for certain types of files. /8

Significance

Browser caching helps by storing some of these files locally in the user's browser. User's first visit to the site will take the same time to load, however when the site is revisited or refreshed the page it already have some of the files they need locally. So no download again the same static files. That indicates the amount of data the browser has to download is now less, and fewer requests need to be made to the server which results decreased in page load times.

Requirements

- CPanel / FTP / SSH access with elevated privileges to modify .htaccess file.
- Enable on mod_headers / mod_expires modules on the server level (Apache).

References

- Cache-Control *(Last updated April 8, 2015)*
- Cache-Control (*Ilya Grigorik, Last updated February 9, 2017)*
- Apache Module mod_headers
- Apache Module mod_expires

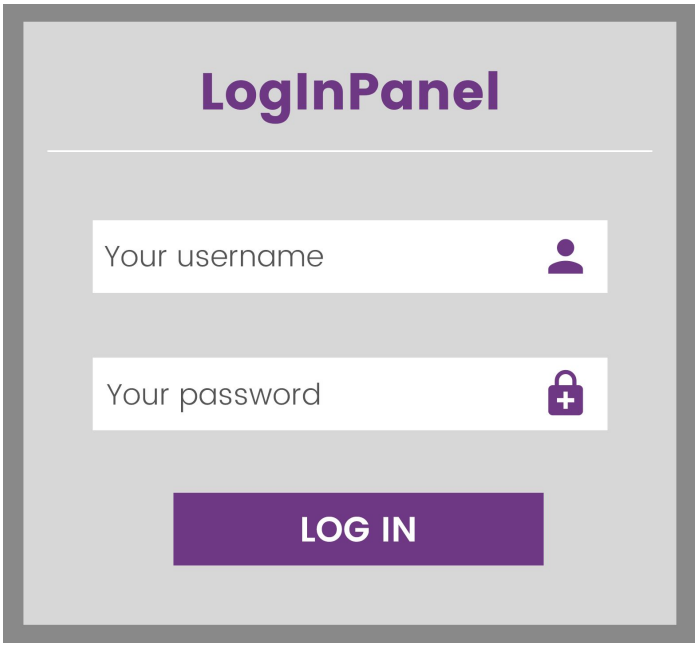
High Level Procedure

1. Login to any panel to access ***.htaccess*** file.
2. Edit the ***.htaccess*** file to add/modify the ***Expire Caching*** code section.
3. Save the changes.

Detail Procedure

Step 1

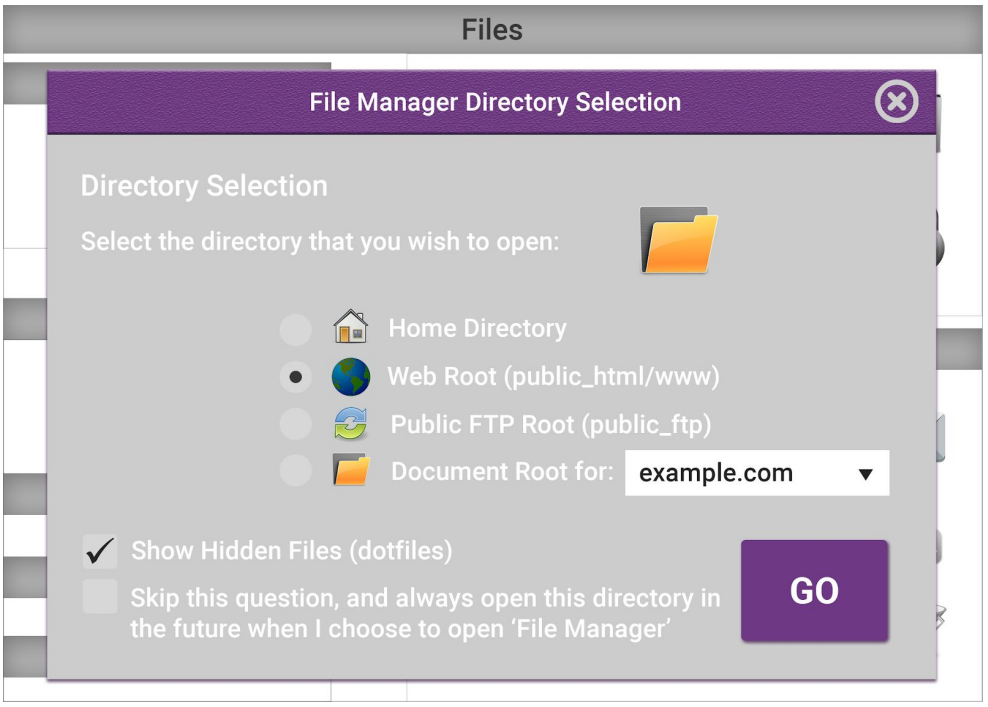
Login to the ***CPanel/FTP*** (or any other available panel) using the login credentials.



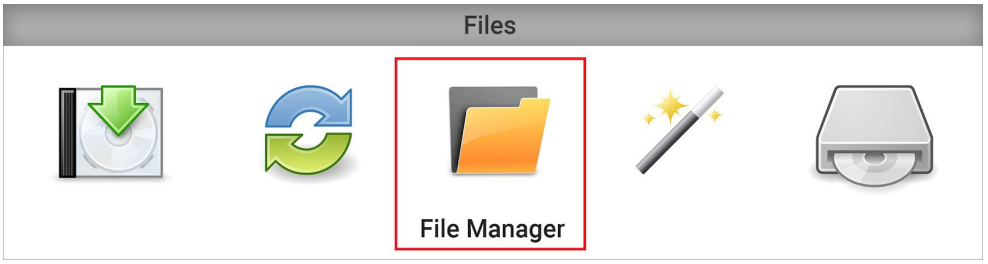
Source: internal mockup

Step 2

In ***CPanel*** (for example), open ***File Manager*** Folder. While opening it may ask for ***Directory Selection***, select ***Web Root (public_html/www***). And make sure that the ***"Show Hidden Files"*** check box is ***marked,*** which will enable ***"View all hidden files"*** on ***CPanel***; in some cases the ***.htaccess*** files will be hidden.



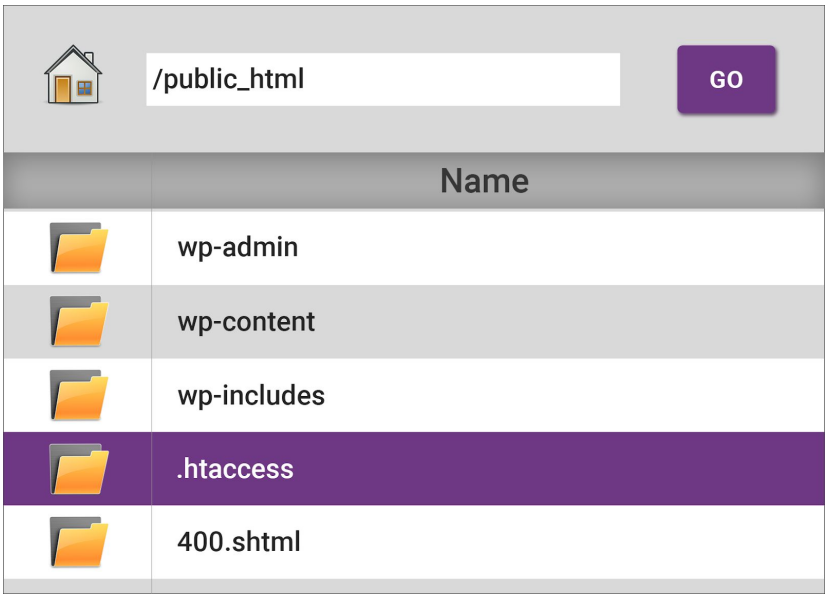
Source: internal mockup



Source: internal mockup

Step 3

In ***File Manager*** folder, look for the ***.htaccess*** file.



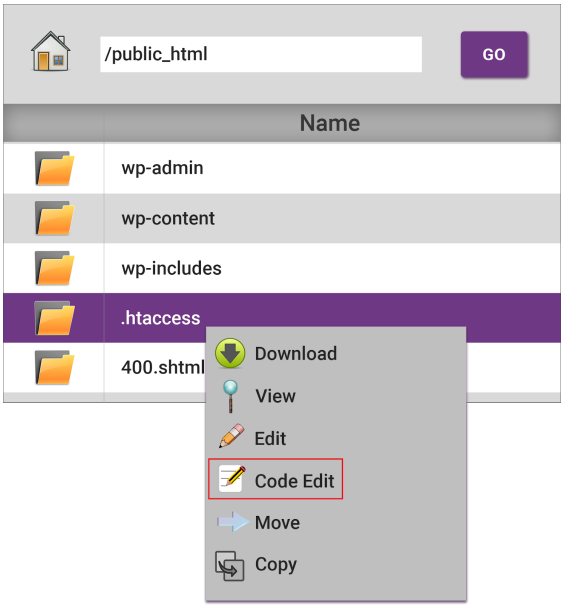
Source: internal mockup

Step 4

Select the ***.htaccess*** file (this is in the root directory in this case). If the ***.htaccess*** file is not available then we can create a new ***.htaccess*** file.

Step 5

Then right click on the ***.htaccess*** and select ***Code Edit***.



Source: internal mockup

Step 6

In ***Code Edit*** editor paste the following code (Please append and avoid replace if possible) after the earlier default code in ***.htaccess***. Below is the sample code that is subject to modification to enable caching on ***.htaccess*** file (below codes can be modified for caching duration/resources).

```
## EXPIRES CACHING ##
<IfModule mod_expires.c>
ExpiresActive On
AddType application/vnd.ms-fontobject .eot
AddType application/x-font-ttf .ttf
AddType application/x-font-opentype .otf
AddType application/x-font-woff .woff
AddType image/svg+xml .svg
ExpiresByType application/vnd.ms-fontobject "access plus 1 year"
ExpiresByType application/x-font-ttf "access plus 1 year"
ExpiresByType application/x-font-opentype "access plus 1 year"
ExpiresByType application/x-font-woff "access plus 1 year"
ExpiresByType image/svg+xml "access plus 1 year"
ExpiresByType image/jpg "access plus 1 year"
ExpiresByType image/jpeg "access plus 1 year"
ExpiresByType image/gif "access plus 1 year"
ExpiresByType image/png "access plus 1 year"
ExpiresByType text/css "access plus 1 month"
ExpiresByType application/pdf "access plus 1 month"
ExpiresByType text/x-javascript "access plus 1 month"
ExpiresByType application/x-shockwave-flash "access plus 1 month"
ExpiresByType image/x-icon "access plus 1 year"
ExpiresDefault "access plus 2 days"
</IfModule>
## EXPIRES CACHING ##

##Caching for mod_headers##
<IfModule mod_headers.c>
   Header set Connection keep-alive
   <filesmatch "\.(ico|flv|jpg|jpeg|png|gif|css|swf|woff)$">
   Header set Cache-Control "max-age=2678400, public"
   </filesmatch>
   <filesmatch "\.(html|htm)$">
   Header set Cache-Control "max-age=7200, private, must-revalidate"
   </filesmatch>
   <filesmatch "\.(pdf)$">
   Header set Cache-Control "max-age=86400, public"
   </filesmatch>
   <filesmatch "\.(js)$">
   Header set Cache-Control "max-age=2678400, private"
   </filesmatch>
</IfModule>
## EXPIRES CACHING ##
```

Source: https://gtmetrix.com/leverage-browser-caching.html

Step 7

Click on ***Save Change*** (mostly near upper right corner in CPanel) and save it.

Step 8

Save the ***.htaccess*** file and then ***refresh*** the webpage in the browser.

Step 9

Perform a round of Smoke Testing to make sure there are zero negative impacts on the website.

Step 10

Check if the changes improve the speed of the website by using the developer tools Page Speed Insight and Web Page Test.

Enabling Browser Caching for IIS

Browser caching is a technique to store some files in the local browser. When a browser loads a webpage it downloads all the web files for properly displaying the page which includes all the images, HTML, CSS, and JavaScript. To enable browser caching, the web server's HTTP headers need to be edited. This needs to be followed by setting the expiration time for certain types of files.
The IIS output cache supports two cache policies:

- User-mode output cache policy, which uses a cache that resides in an IIS worker process.
- Kernel-mode cache policy, which uses a cache that resides in http.sys, a kernel-mode driver.

Significance

Browser caching helps by storing some of these files locally in the user's browser. User's first visit to the site will take the same time to load, however when the site is revisited or refreshed the page it already have some of the files they need locally. So no download again the same static files. That indicates the amount of data the browser has to download is now less, and fewer requests need to be made to the server which results decreased in page load times.

Requirements

- CPanel / FTP / RDP access with elevated privileges to modify **web.config file**.
- Applicable for version IIS7 and above.
- Permission from hosting provider to enable Caching.

References

- Leverage Browser Caching *(Last updated April 8, 2015)*
- Cache-Control (*Ilya Grigorik, Last updated February 9, 2017)*
- Configure IIS 7 Output Caching
- Caching <caching>

High Level Procedure 1 (Manual)

1. Login to any panel to access ***web.config*** file.
2. Edit the ***web.config*** file to add/modify the ***Expire Caching*** code section.
3. Save the changes.

Detail Procedure 1 (Manual): Web.config (Application Level)

Step 1

Login to the ***CPanel/FTP/RDP*** with the customer's login credentials.

Step 2

Locate ***Root Folder*** on the site.

Step 3

Enable ***"View all hidden files"*** on the ***FTP/Cpanel***. In some cases the ***web.config file*** *(not* ***.htaccess****)*, will be hidden.

Step 4

Locate the ***web.config file***; if the ***web.config*** file is not available then we can create a new configuration file.

Step 5

Below is an example configuration code that will be personalized to enable caching on ***web.config file***. In ideal conditions the **Web.config** file will be available in the *Root Folder* of the site.

```
## EXPIRES CACHING ##
<staticContent>
  <clientCache cacheControlMode="UseMaxAge" cacheControlMaxAge="7.00:00:00" />
</staticContent>
<caching enabled="true" enableKernelCache="true">
<profiles>
  <add extension=".asp" policy="CacheUntilChange"
kernelCachePolicy="CacheUntilChange" />
  <add extension=".aspx" policy="CacheUntilChange"
kernelCachePolicy="CacheUntilChange" />
  <add extension=".css" policy="CacheUntilChange"
kernelCachePolicy="CacheUntilChange" />
  <add extension=".js" policy="CacheUntilChange"
kernelCachePolicy="CacheUntilChange" />
  <add extension=".xml" policy="CacheUntilChange"
kernelCachePolicy="CacheUntilChange" />
  <add extension=".svg" policy="CacheUntilChange"
kernelCachePolicy="CacheUntilChange" />
  <add extension=".html" policy="CacheUntilChange"
kernelCachePolicy="CacheUntilChange" />
  <add extension=".jpg" policy="CacheUntilChange"
kernelCachePolicy="CacheUntilChange" />
  <add extension=".png" policy="CacheUntilChange"
kernelCachePolicy="CacheUntilChange" />
  <add extension=".jpeg" policy="CacheUntilChange"
kernelCachePolicy="CacheUntilChange" />
  <add extension=".gif" policy="CacheUntilChange"
kernelCachePolicy="CacheUntilChange" />
  <add extension=".ico" policy="CacheUntilChange"
kernelCachePolicy="CacheUntilChange" />
  <add extension=".ttf" policy="CacheUntilChange"
kernelCachePolicy="CacheUntilChange" />
  <add extension=".php" policy="CacheUntilChange"
kernelCachePolicy="CacheUntilChange" />
</profiles>
```

Source: https://www.iis.net/configreference/system.webserver/caching?showTreeNavigation=true

Step 6

Save the ***web.config file*** and then refresh your webpage on the opened browser.

Step 7

Perform a round of “Smoke Testing” to make sure there are zero negative impacts on the site.

Step 8

Make sure that the caching mechanism is working with the developer tool or any third party tools (PSI, GTmetrix etc.).

Detail Procedure 2 (With GUI)

Step 1

From the Start menu, click ***Administrative Tools***, and then click ***Internet Information Services*** (IIS) Manager.

Step 2

In the tree view on the left side, find ***your application***.

Step 3

Select the ***Output Caching menu item***.

Step 4

In the right column, click ***Add in the Action menu***. You can add your output caching rule here.

Step 5

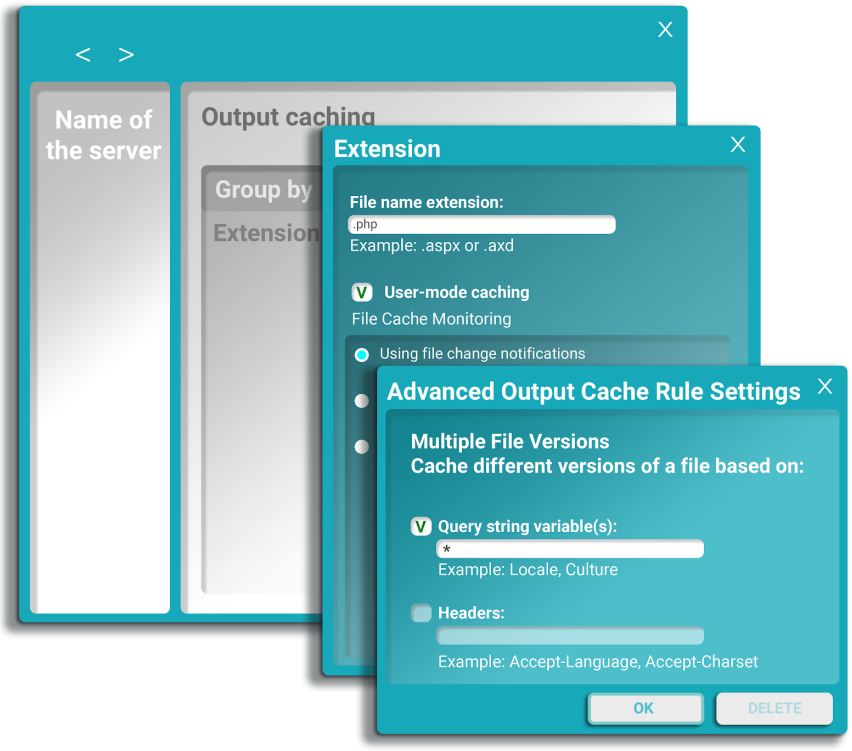
In the File name extension field, for example, type ***.php***, and then select User-mode caching.

Step 6

Click Advanced, and then select the Query string variable(s) check box.

Step 7

Enter the appropriate variable(s) in the Query string variable(s) text box.



Source: internal mockup

Step 8

Check the website post changes for improved speed of the mobile website with Page Speed Insight & Web Page Test.

Enabling Browser Caching for Nginx

Browser caching is a technique to store some files in local browser. When a browser loads a webpage it downloads all the web files for properly displaying the page which includes all the images, HTML, CSS, and JavaScript. To enable browser caching, the web server's HTTP headers need to be edited. This needs to be followed by setting the expiration time for certain types of files.

Significance

Browser caching helps by storing some of these files locally in the user's browser. User's first visit to the site will take the same time to load, however when the site is revisited or refreshed the page it already have some of the files they need locally. So no download again the same static files. That indicates the amount of data the browser has to download is now less, and fewer requests need to be made to the server which results decreased in page load times.

Requirements

SSH access setup for root or a ***sudo*** user.

References

- Leverage Browser Caching (*Last updated April 8, 2015)*
- Cache-Control (*Ilya Grigorik, Last updated February 9, 2017)*
- Module ngx_http_headers_module
- https://www.howtoforge.com/make-browsers-cache-static-files-on-nginx

High Level Procedure

1. Login to the Nginx web server.
2. Edit the ***nginx config*** file to add/modify the "Browser Caching config" section.
3. Save the changes.

Detail Procedure

Step 1

Login to the root file using ***SSH Details***.

Step 2

Locate the website configuration file. Usually, the configuration file will be located in ***/etc/nginx/sites-available/ or /etc/nginx/sites-enabled / in similar folder*** and take required backup of ***nginx config file***.

Step 3

Open and edit the configuration file using ***Vi / Nano*** or any other editing command.

Step 4

The Expired HTTP header can be set with the help of the expired directive which can be placed inside **http {}, server {}, location {},** or an *if statement* inside a **location {} block**. Usually you will use it in a location block for your static files. Please check the example configuration code below.

```
# Browser Caching config

location ~*  \.(jpg|jpeg|png|gif|ico|css|js)$ {
        expires 365d;
    }
location ~*  \.(pdf)$ {
        expires 30d;
}
```

Source: https://gist.github.com/Antoinebr/c43145e598d60bf954cbf2657b689920

Step 5

Save the ***nginx config* file** using ***":wq"*** command (If using VIM editor)

Step 6

Restart the ***Nginx server*** using the command ***"sudo service nginx restart."***

Step 7

Check the website post changes for improved speed of the mobile website with Page Speed Insight & Web Page Test.

Enable Compression for Apache

This technique is used to compress the http request to reduce the size of the transferred responses. Compression is enabled via configuration files and depends on the type of web server (Apache in this case).

Significance

By using this technique we can significantly reduce the amount of time to download the resource, reduce data usage for the client, and improve the time to the first render of your pages.

Requirements

- CPanel / FTP / SSH access with elevated privileges to modify the .htaccess file.
- Compression modules (mod_gzip or mod_deflate) will need to be enabled on the server level (Apache). There are two gzip modules (mod_gzip or mod_deflate) in apache server, need to verify with hosting account which module has been used and enable it. And then ask for suggestions which .htaccess code to use.

References

- Enable Compression (*Last updated April 8, 2015)*
- Apache Module mod_deflate
- Enable gzip compression

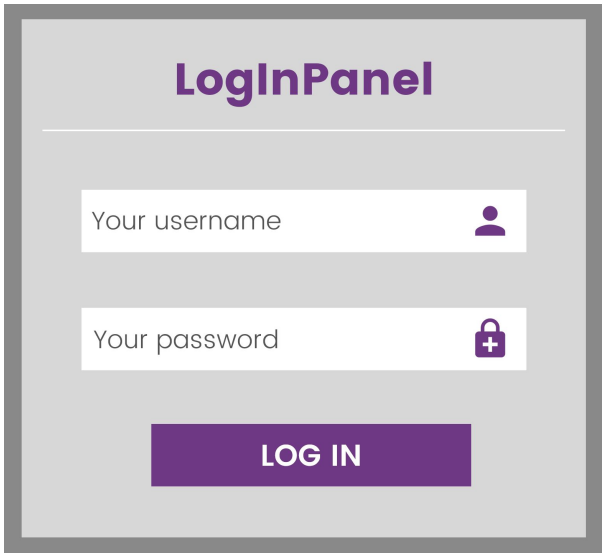
High Level Procedure

1. Login to the Apache web server
2. Edit the ***.htaccess*** file to add/modify the “Compression” section to enable compression.
3. Save the changes.

Detail Procedure

Step 1

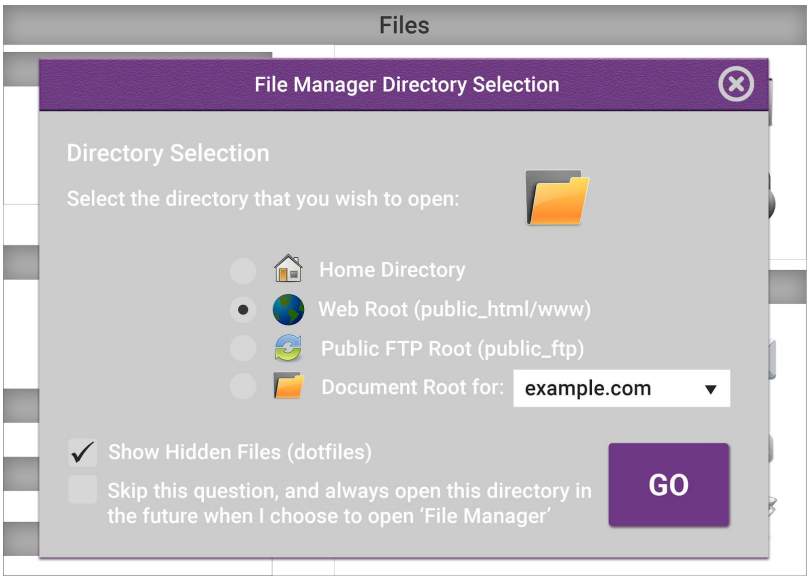
Login to their ***CPanel/FTP*** using the advertiser's login credentials.



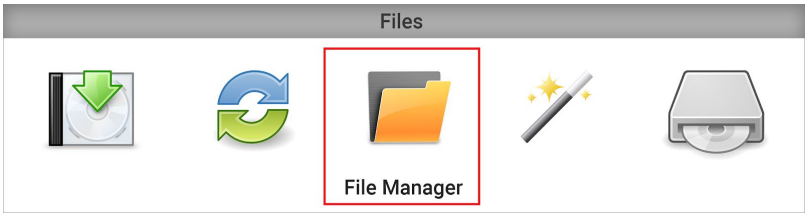
Source: internal mockup

Step 2

In ***CPanel***, open ***File Manager*** Folder. While opening it may ask for ***Directory Selection***, select ***Web Root (public_html/www***). And make sure that the ***"Show Hidden Files"*** check box is ***marked,*** which will enable **"View all hidden files"** on ***CPanel***; in some cases the ***.htaccess*** files will be hidden.



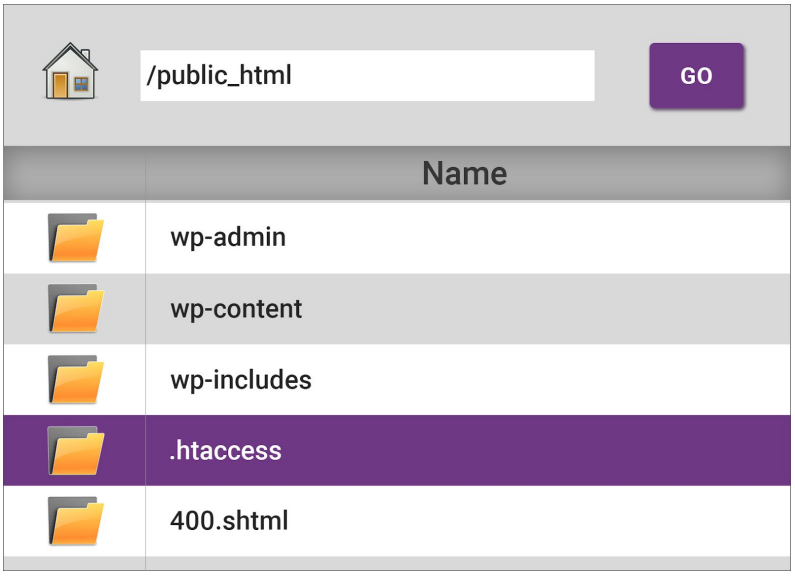
Source: internal mockup



Source: internal mockup

Step 3

In ***File Manager*** folder, look for ***.htaccess*** file.



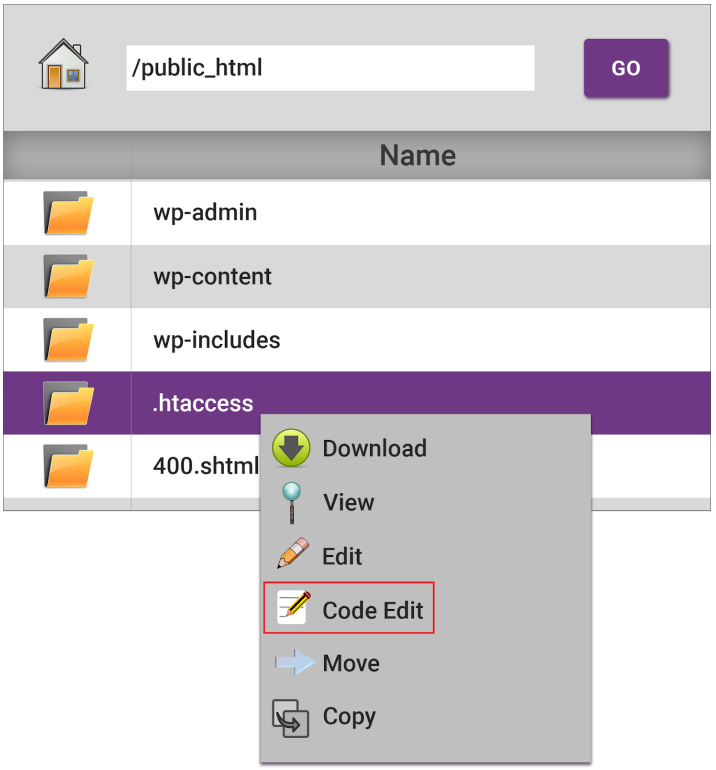
Source: internal mockup

Step 4

Select the ***.htaccess*** file (this is in root directory). If ***.htaccess*** file is not available then we can create a new one, ***.htaccess.***

Step 5

Then right click ***.htaccess*** and select ***Code Edit***.



Source: internal mockup

Step 6

Below is the example configuration code to be modified in order to **enable compression** on the ***.htaccess file*** (The below code can be modified for caching duration/resources).

For mod_deflate Module

```
<IfModule mod_deflate.c>
# Compress HTML, CSS, JavaScript, Text, XML and fonts
AddOutputFilterByType DEFLATE application/javascript
AddOutputFilterByType DEFLATE application/rss+xml
AddOutputFilterByType DEFLATE application/vnd.ms-fontobject
AddOutputFilterByType DEFLATE application/x-font
AddOutputFilterByType DEFLATE application/x-font-opentype
AddOutputFilterByType DEFLATE application/x-font-otf
AddOutputFilterByType DEFLATE application/x-font-truetype
AddOutputFilterByType DEFLATE application/x-font-ttf
AddOutputFilterByType DEFLATE application/x-javascript
AddOutputFilterByType DEFLATE application/xhtml+xml
AddOutputFilterByType DEFLATE application/xml
AddOutputFilterByType DEFLATE font/opentype
AddOutputFilterByType DEFLATE font/otf
AddOutputFilterByType DEFLATE font/ttf
AddOutputFilterByType DEFLATE image/svg+xml
AddOutputFilterByType DEFLATE image/x-icon
AddOutputFilterByType DEFLATE text/css
AddOutputFilterByType DEFLATE text/html
AddOutputFilterByType DEFLATE text/javascript
AddOutputFilterByType DEFLATE text/plain
AddOutputFilterByType DEFLATE text/xml

# Remove browser bugs (only needed for really old browsers)
BrowserMatch ^Mozilla/4 gzip-only-text/html
BrowserMatch ^Mozilla/4\.0[678] no-gzip
BrowserMatch \bMSIE !no-gzip !gzip-only-text/html
Header append Vary User-Agent
</IfModule>
```

Source:
https://help.blacknight.com/hc/en-us/articles/212523089-GZIP-Compression-Deflate-Linux-Shared-Hosting

First try to enable GZIP, if that fails, use DEFLATE. If both fail, you need to enable the gzip module inside "php.ini".

For mod_gzip Module

```
<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>
```

Source: https://varvy.com/pagespeed/enable-compression.html

Step 7

Click on Save Change (mostly near upper right corner) and save it.

Step 8

Save the ***.htaccess*** *file* and then **refresh** the webpage on the browser.

Step 9

Perform a round of *Smoke Testing* to solidify zero negative impacts on the website.

Step 10

Check the website post changes for improved speed of the mobile website with Page Speed Insight & Web Page Test.

Enable Compression for IIS

This technique is used to compress the http request to reduce the size of the transferred responses. Compression is enabled via configuration files and depends on the type of web server (IIS in this case).

Significance

By using this technique we can significantly reduce the amount of time to download the resource, reduce data usage for the client, and improve the time to first render of your pages.

Requirements

- **CPanel / FTP / RDP** access with elevated privileges to modify the **web.config file**.
- Needs to be treated for IIS7 and above versions. (*IIS6 users or early versions please note: We'll need the advertiser to enable a few additional modules from the server level; this will need to be handled by the hosting provider for we cannot complete this task since we won't be accessing the server on the treatment call. Please complete this prior to the call.*).
- Permission from the hosting provider to enable Caching.

References

- Enable Compression *(Last updated April 8, 2015)*
- HTTP Compression <httpCompression>

High Level Procedure

1. Login to the IIS web server
2. Edit the ***web.config*** file to add/modify the "Compression" section to enable compression.
3. Save the changes.

Detail Procedure: Web.config (Application Level)

Step 1

Login to the ***CPanel/FTP/****RDP* using the provided credentials.

Step 2

Locate the ***Root Fold****er* of the site.

Step 3

Create a new *txt file* with dummy data to verify/confirm the root of the site (Once the *Root Folder* is identified, the txt file can be removed).

Step 4

Enable ***"View all hidden files"***on *FTP/Cpanel*. In some cases the ***web.config*** *file* will be hidden.

Step 5

Locate ***web.config*** *file* (If ***web.config*** *file* is not available then we can create a new ***config file***).

Step 6

Below is the sample configuration code to be updated/modified for the ***web.config*** *file* (In the standard format, the ***web.config*** *file* will be available in the root folder of the site).

```
     <httpCompression  directory="%SystemDrive%\inetpub\temp\IIS  Temporary
Compressed Files">
      <scheme name="gzip" dll="%Windir%\system32\inetsrv\gzip.dll"/>
      <dynamicTypes>
        <add mimeType="text/*" enabled="true"/>
        <add mimeType="message/*" enabled="true"/>
        <add mimeType="application/javascript" enabled="true"/>
        <add mimeType="*/*" enabled="false"/>
      </dynamicTypes>
      <staticTypes>
        <add mimeType="text/*" enabled="true"/>
        <add mimeType="message/*" enabled="true"/>
        <add mimeType="application/javascript" enabled="true"/>
        <add mimeType="*/*" enabled="false"/>
      </staticTypes>
    </httpCompression>
                          <urlCompression       doStaticCompression="true"
doDynamicCompression="true"/>
```

Source:
https://www.iis.net/configreference/system.webserver/urlcompression?showTreeNavigation=true

Step 7

Save the ***web.config*** *file* and then refresh your webpage in the browser.

Step 8

Perform a round of *Smoke Testing* to make sure zero negative impacts on the site.

Step 9

Check the website post changes for improved speed of the mobile website with Page Speed Insight & Web Page Test.

Enable Compression for Nginx

This technique is used to compress the http request to reduce the size of the transferred responses. Compression is enabled via configuration files and depends on the type of web server (Nginx in this case).

Significance

By using this technique we can significantly reduce the amount of time to download the resource, reduce data usage for the client, and improve the time to first render of your pages.

Requirement

- ***SSH access* setup** for root or a pseudo user.
- **ngx_http_gzip_module** or related compression module to be enabled

References

- Enable Compression *(Last updated April 8, 2015)*
- Module ngx_http_gzip_module

High Level Procedure

1. Login to the Nginx web server.
2. Edit the ***nginx config*** file to add/modify the "Compression (Gzip Config)" section to enable compression.
3. Save the changes.

Detail Procedure

Step 1

Login to the ***Root File*** using ***SSH*** details.

Step 2

Locate the ***Nginx Configuration file***. Usually, configuration file will be located in the ***/etc/nginx/sites-available/***.

Step 3

Open and edit the ***Nginx Configuration file*** using ***Vi / Nano*** or any other editing command.

Step 4

The below is an example configuration that'll be modified and updated in order to enable Gzip compression on ***Nginx config file*** (if below gzip code already exist in ***nginx.conf file***, but is either commented out using a ***#*** or has different values assigned to them. If the lines are commented out, we may either uncomment them and match the values, or simply add the ***new gzip*** values to the file).

```
# Gzip Config
gzip on;
gzip_types text/plain text/css application/json application/javascript
text/xml application/xml application/xml+rss text/javascript;
```

Source: https://gist.github.com/Antoinebr/c43145e598d60bf954cbf2657b689920

Step 5

Save the ***nginx.config*** **file** using ***":wq"***.

Step 6

Restart the ***Nginx server*** using **"sudo service nginx restart".**

Step 7

Check the website post changes for improved speed of the mobile website with Page Speed Insight & Web Page Test.

Avoiding Loading Duplicate JS & CSS Files

Sometimes some resources (JS, CSS) get loaded more than once.

Significance

This takes up page weight and slows down the loading time. This can happen when the same CSS or JS file is requested from multiple locations. For example, sometimes two different scripts load jQuery. To recheck for double loading files: ***Chrome DevTools > Network > All > Filter by Name.***

References

Preventing javascript files from loading multiple times

High Level Procedure

1. Open chrome Dev Tool and go to *Network > All > Filter by Name*
2. Identify the duplicate CSS & JS files
3. Remove the duplicate CSS & JS files and save the changes.

Detail Procedure

Step 1

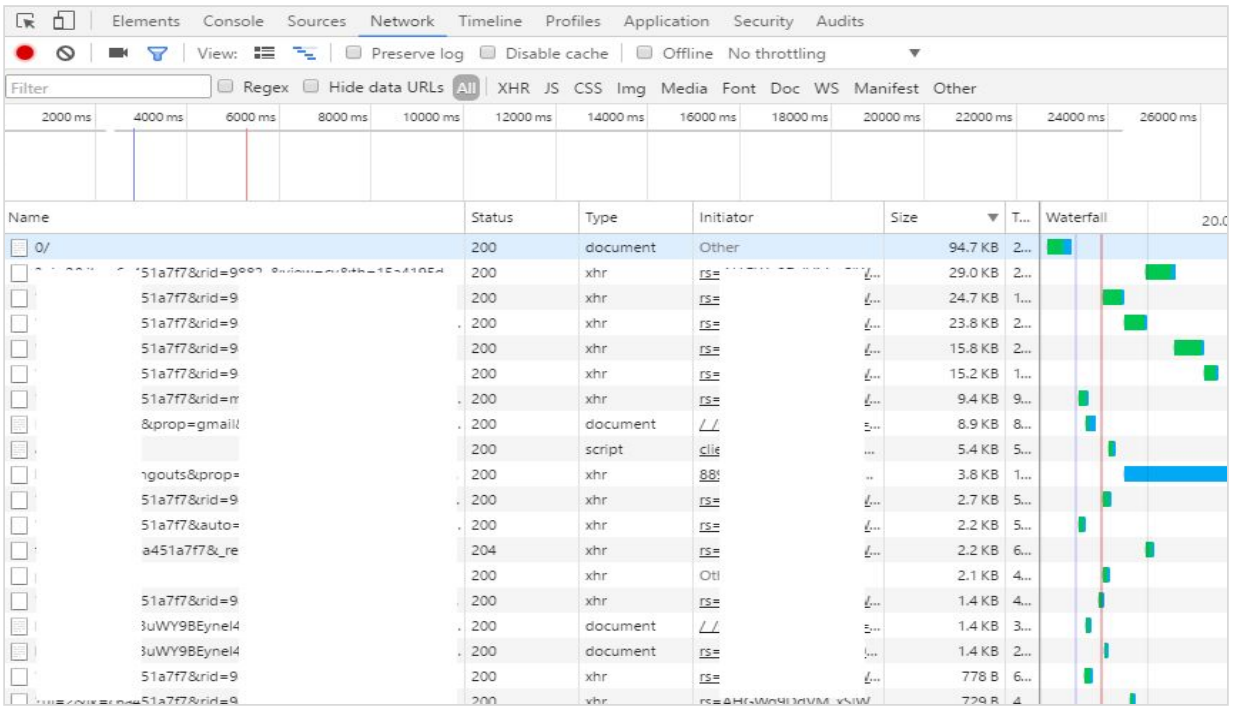
Open the mobile website.

Step 2

Open the Chrome Dev Tool

Step 3

Go to ***Network > All > Filter by Name***



Source: Internal Lab Experiment in Google

Step 4

Identify the duplicate files (CSS or JS file)

Step 5

List down the duplicate files (CSS or JS file)

Step 6

Remove the duplicate files (CSS or JS file)

Step 7

Check the website post changes for improved speed of the mobile website with Page Speed Insight & Web Page Test.

Image Optimization - Removing Duplicate Images

Sometimes an image gets loaded more than once. This takes up page weight and slows download time.

Significance

This takes up page weight and slows down load time. This can happen when the same image file is requested from multiple locations or multiple times. To recheck for double loading files: ***Chrome DevTools > Network > All > Filter by Name.***

References

- Optimize Images (*Last updated December 14, 2016)*
- Image Optimization *(Ilya Grigorik, Last updated February 9, 2017)*

High Level Procedure

1. Open Page Speed Insight (PSI) and run the website and download the optimized images from Page Speed Insight. Analyse for duplicate images.
2. Open chrome Dev Tool and go to ***Network > All > Filter by Name***, and look for if duplicate images need to be removed.
3. Replace the images and save the changes.

Detail Procedure

Step 1

Open Page Speed Insight.

Step 2

Enter the mobile website in Page Speed Insight and click test.

Step 3

Look for images to be optimized and note down the list of images and download the optimized images.

Source: Page Speed Insight

Step 4

Open Chrome Dev Tool

Step 5

Go to ***Network > All > Filter by Name***

Step 6

Identify the duplicate images

Step 7

List down the duplicate images.

Step 8

Remove the duplicate images

Step 9

Check the website post changes for improved speed of the mobile website with Page Speed Insight & Web Page Test.

Image Optimization - Compress Images

Many times some of the image's sizes are very large and due to the big sizes they take long time to download which can cause increase in page load time.

Vector vs. Raster images:

Vector formats are ideally suited for images that consist of simple geometric shapes (for example, logos, text, icons, and so on), and deliver sharp results at every resolution and zoom setting, which makes them an ideal format for high-resolution screens and assets that need to be displayed at varying sizes.

Raster graphics represent an image by encoding the individual values of each pixel within a rectangular grid. Raster images do not have the same nice properties of being resolution or zoom independent - when you scale up a raster image you'll see jagged and blurry graphics. As a result, you may need to save multiple versions of a raster image at various resolutions to deliver the optimal experience to your users.

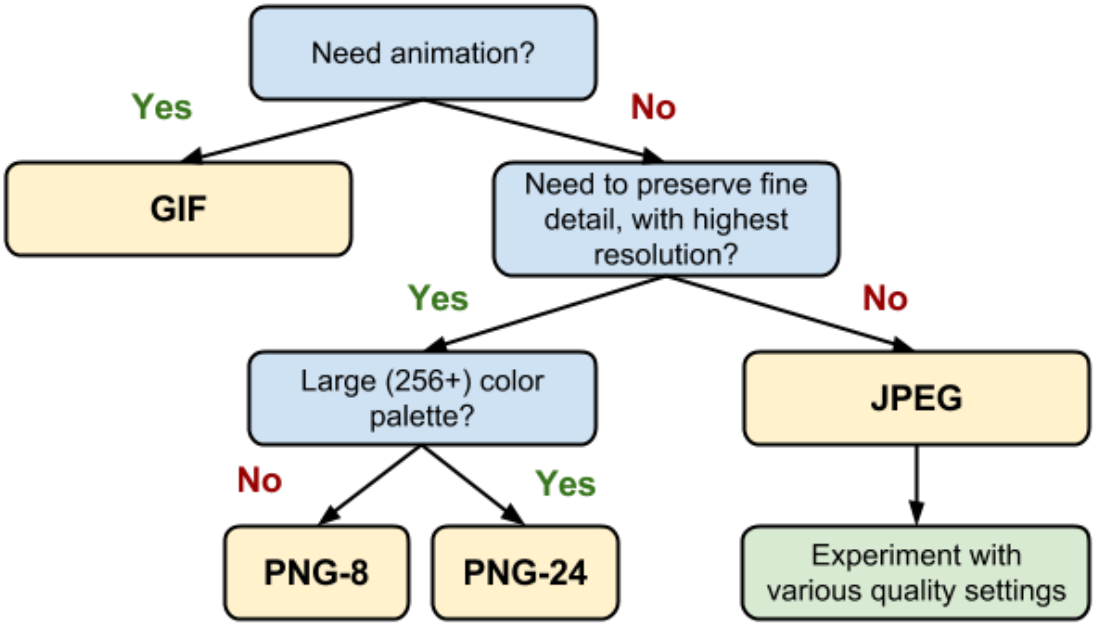
Lossless vs lossy image compression:

Image is processed with a "lossy" filter that eliminates some pixel data, and image is processed with a "lossless" filter that compresses the pixel data. For optimal configuration of lossy and lossless optimization, it depends on the image contents and your own criteria such as the tradeoff between ***filesize*** and ***artifacts*** introduced by lossy compression: in some cases you may want to skip lossy optimization to communicate intricate detail in its full fidelity, and in others you may be able to apply aggressive lossy optimization to reduce the filesize of the image asset. This is where your own judgment and context need to come into play - there is no one universal setting.
In addition to different lossy and lossless compression algorithms, different image formats support different features such as animation and transparency (alpha) channels. As a result, the choice of the "right format" for a particular image is a combination of desired visual results and functional requirements.

Format	Transparency	Animation	Browser
GIF	Yes	Yes	All
PNG	Yes	No	All
JPEG	No	No	All
JPEG XR	Yes	Yes	IE
WebP	Yes	Yes	Chrome, Opera, Android

Source:
https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/image-optimization

The following diagram can help to select the right picture format for the purpose:



Source:
https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/image-optimization

Significance

Larger the image, longer will be to download and results longer load time. We need to adjust the image to the lowest file size acceptable while keeping an eye out for image quality. This involves some manual intervention to change the file colors (in ***.gif and .png***) and quality percentage on ***.jpgs***. A general rule of thumb that we may follow is: for banners or header images, a file size up to 60KB is acceptable, and for high end photographs, we try to keep them to within 100KB at an absolute maximum. Analyse and download images that needs to be optimized.

References

- "Optimize Images | PageSpeed Insights | Google Developers." Google Developers. Google Inc., 08 Apr. 2016. Web. 04 Nov. 2016.
- "PageSpeed: Optimize Images." Optimize Images. GT Metrix, 05 Jan. 2016. Web. 07 Nov. 2016.
- Grigorik, Ilya. "Image Optimization | Web | Google Developers." Google Developers. Google Inc., 04 Nov. 2016. Web. 07 Nov. 2016.

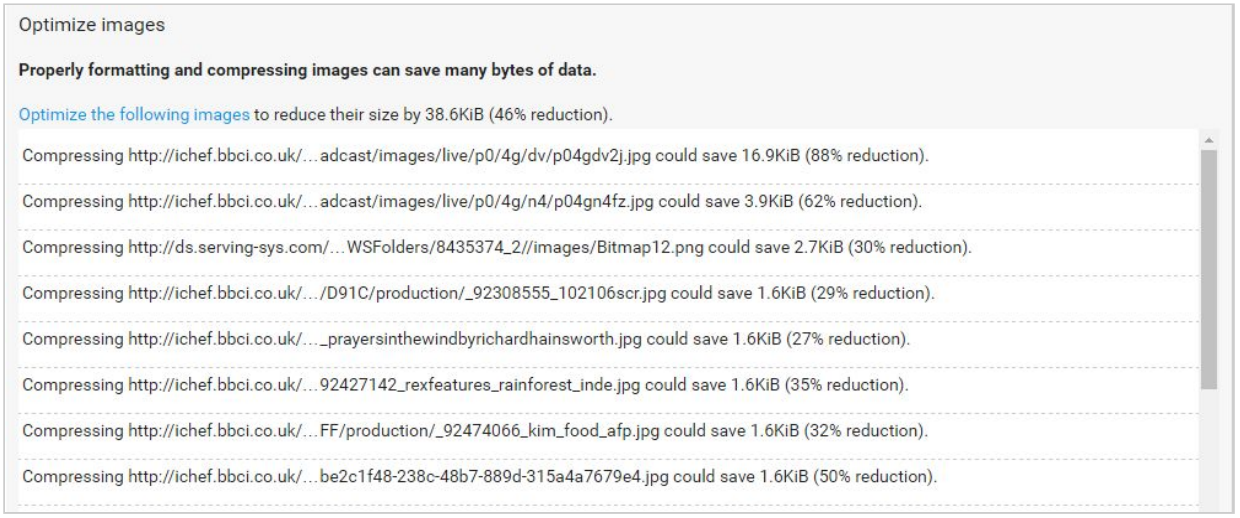
High Level Procedure

1. Open Page Speed Insight and run the website.
2. Download the optimized images from Page Speed Insight.
3. Open chrome Dev Tool and go to ***Network > All > Filter by Name*** and look for if any additional images need to be optimized, and optimize the images as required.
4. Avoid using images that are still>100KB after compression.
5. Save the changes.

Detail Procedure

Step 1

Use Page Speed Insight to get the images detailed required for optimization.
e.g.:



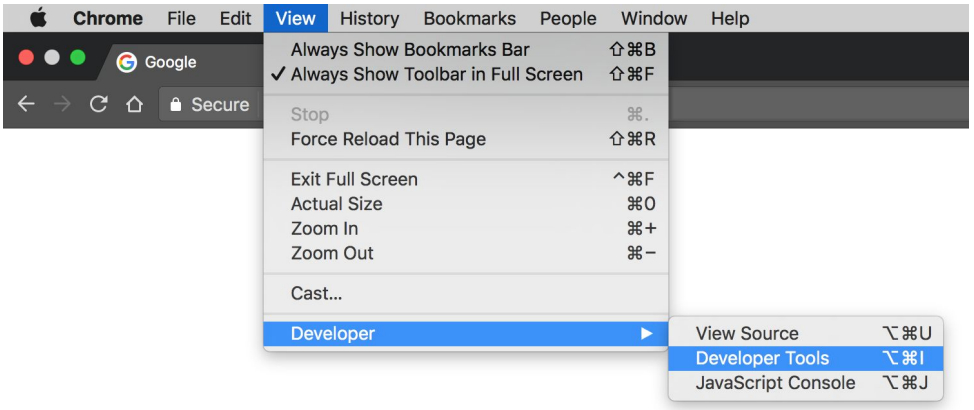
Source: Internal Experiment at Google

Step 2

Download the optimized images from the help of PSI.

Step 3

Then open Chrome Developer tool
e.g.:

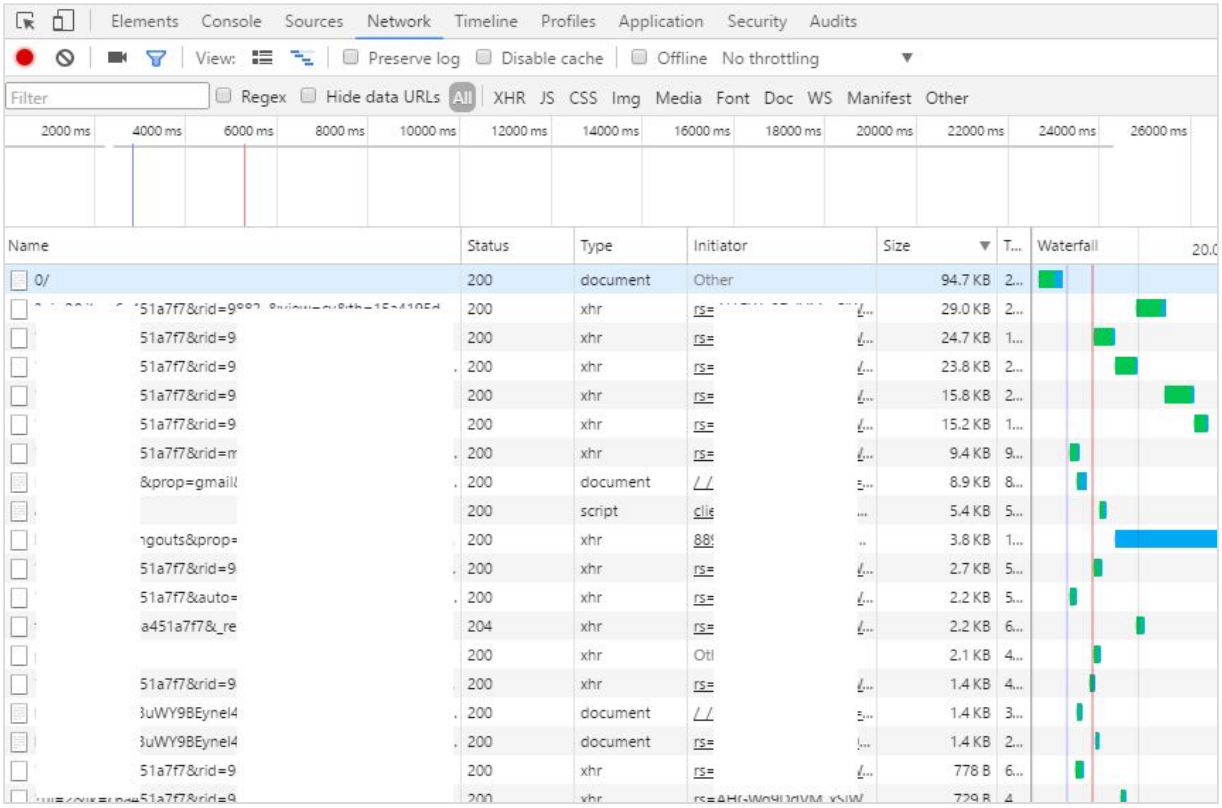


Source: Internal Experiment at Google

Step 4

Go to Network tab

e.g.:



Source: Internal Experiment at Google

Step 5

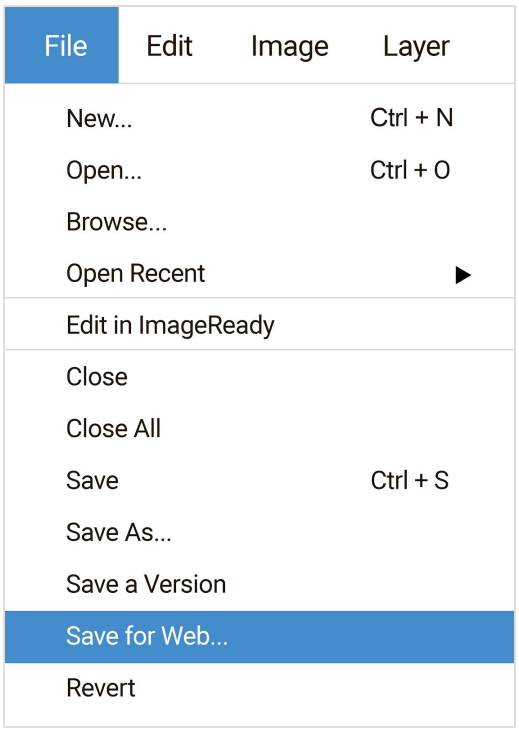
Sort by “Size” and look for big volume images. Also cross check with downloaded images (from PSI) to make sure that no high volume image is missed. If any big image is not captured by PSI then use Photoshop (this is licenced tool!)/any other tools to optimize the images.

Sub Step 5.1

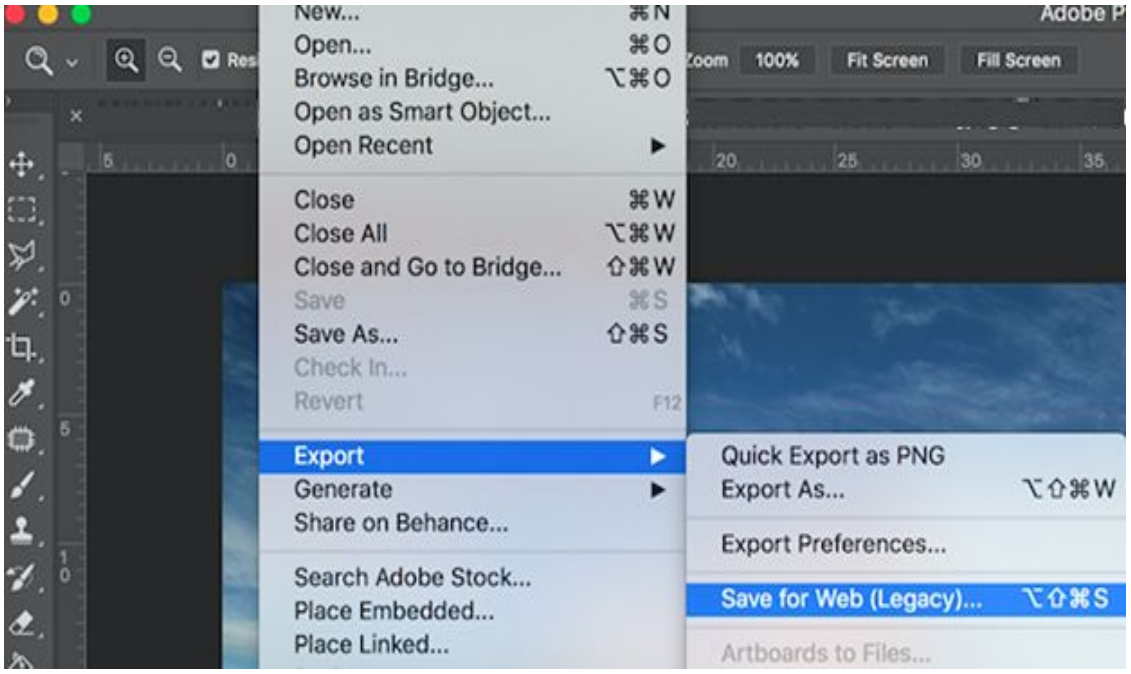
Photoshop is used when the other tools are not able to identify all the images that can be optimized or identify images that can be resized.

e.g.:

Source : How to Use the Photoshop Save for Web Tool (for latest versions)



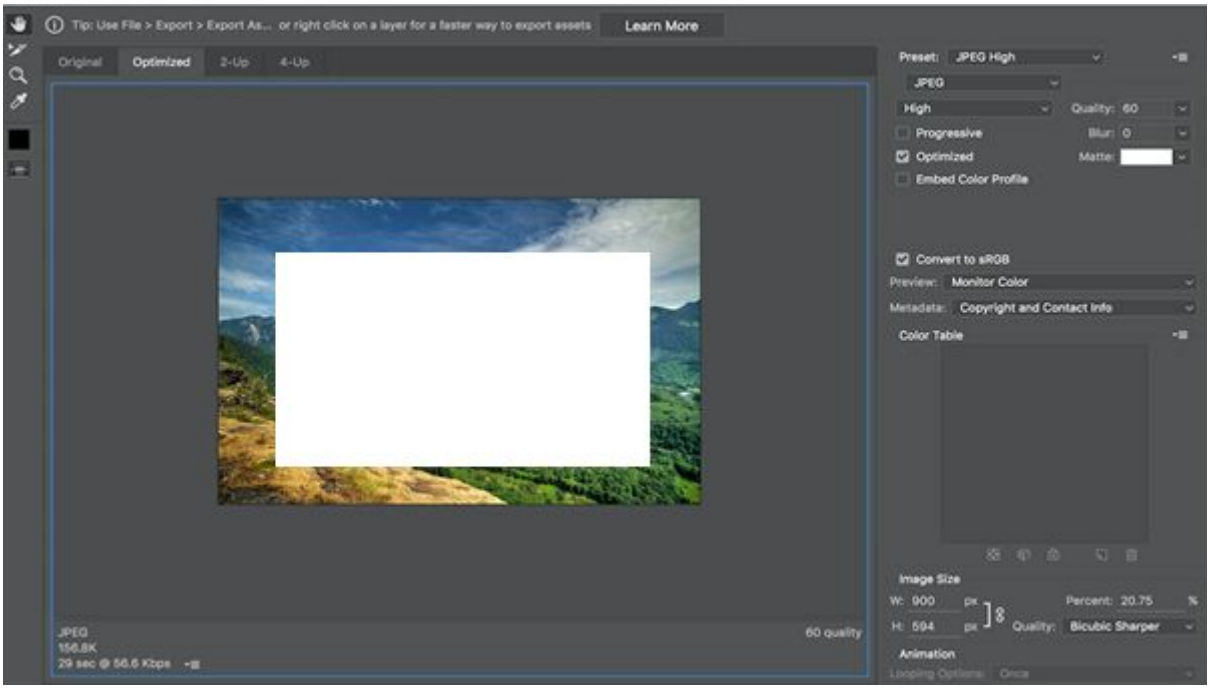
For Older version it should show as below:



Source: Internal Experiment at Google

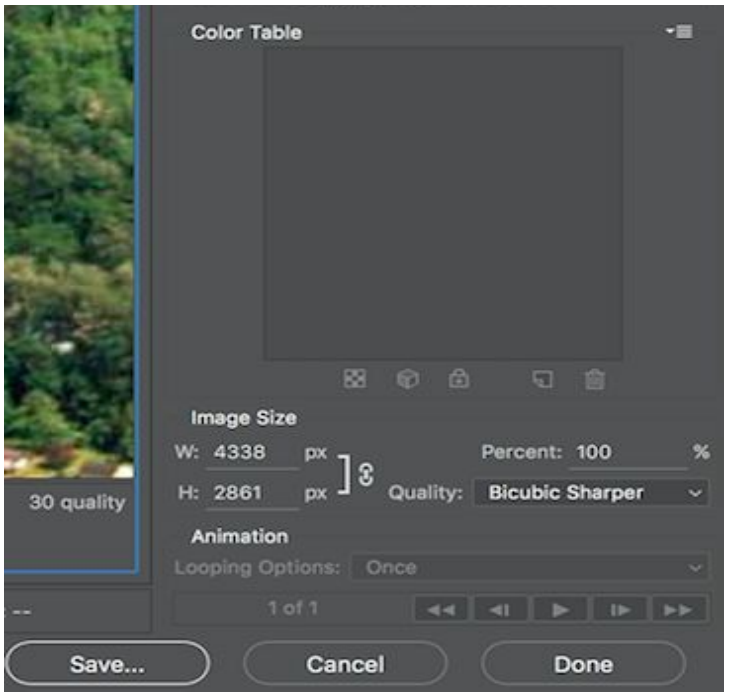
Photoshop allows us to choose the file type for the image and the quality for the image as shown in the screenshots below on the top right hand corner of the

provided screenshot. Ex: JPG in the screenshot below. We'll only use ***JPG, PNG and GIF*** files for treatments. We will use the images that are available of the website and do not change formats to make sure there are no negative impact on the site and to reduce operational issues.
e.g.:



Source: Internal Experiment at Google

The Image sizes can be adjusted at the bottom right hand corner, highlighted in the below screenshot. You can adjust this to the screen size depending on the device used.



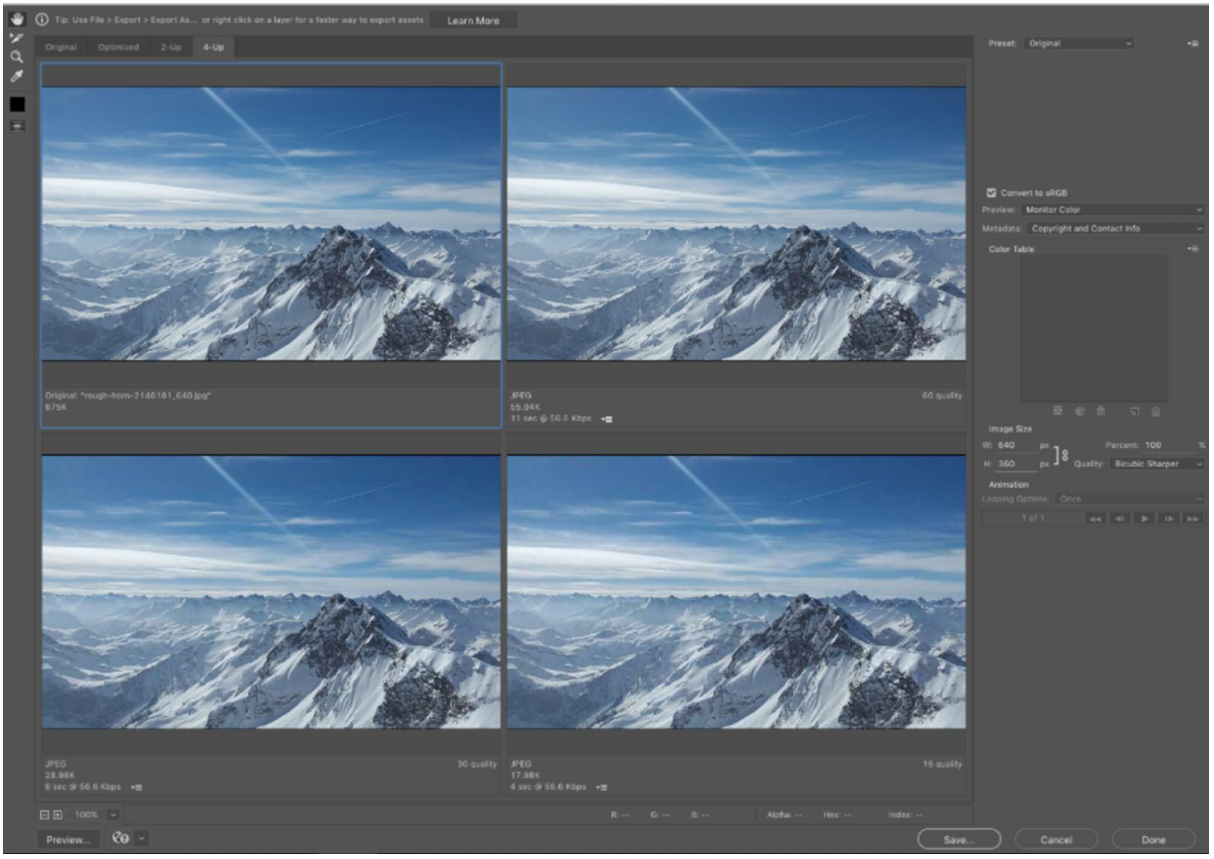
Source: Internal Experiment at Google

Note:
Photoshop also provides an option to compare 3 versions of images(size adjustable) with an original image at once using “4 -Up” panel.

Using which one can decide how far one can go in the quality loss by comparing 3 different compressed image versions with the original image.

Each version gives us the size of the compressed image and the time(in secs) it can take to download at certain speed of the network.

Eg:



Source: Internal experiment at Google

Sub Step 5.2

Alternate tools

In the absence of PhotoShop, these following alternate third party web tools can be used to reformat and resize images with the suggestions above. Google is not recommending these tools directly, but these are some alternative tools available that can be explored based on your judgments.

- Page Speed Insight: It allows to download most required optimized images (in some cases you will not be able to download all the recommended optimised images)
- GTmetrix: It also offer to download optimized images.
- http://www.imageoptimizer.net/

- https://kraken.io/web-interface

Step 6

Images on the site might be located through multiple image folders. So all optimized version of images have to be hosted locally according to the live image folder structure.

Step 7

Check the website post changes for improved speed of the mobile website with Page Speed Insight & Web Page Test.

*Note 1

Image folder structure will not be same for all sites (e.g.: WordPress will auto create a new folder depending on the date and time of the upload)

*Note 2

In some cases the tools will not be able to identify all the images that can be optimized or identify images that can be resized. In such cases the developer will be able to identify images that can be resized or compressed by checking the web pages (e.g.: If the website is resizing a ***800X500 px*** image to ***300X200 px*** using CSS then we will recommend to crop / optimize the images using Photoshop).

Minify CSS

Many times the size of the CSS files are huge and we can reduce the size significantly by minifying it, means by removing the unnecessary spaces between codes.

Significance

Minify helps to reduce the size of the CSS file results less load on network and faster load time.

Requirement

CPanel / FTP access with elevated privileges to add / modify CSS file.

References

- "Optimize CSS Delivery | PageSpeed Insights | Google Developers." Google Developers. Google Inc., *26 Apr. 2016. Web. 04 Nov. 2016.*
- Esposito, Dino. "Cutting Edge - Programming CSS: Bundling and Minification." Microsoft.com, *01 Oct. 2013. Web. 04 Nov. 2016*.

High Level Procedure

1. Identify the CSS files to minify.
2. Minify the CSS files (using any tool of the choice) and replace with original CSS files.
3. Save the changes.

Detail Procedure

Step 1

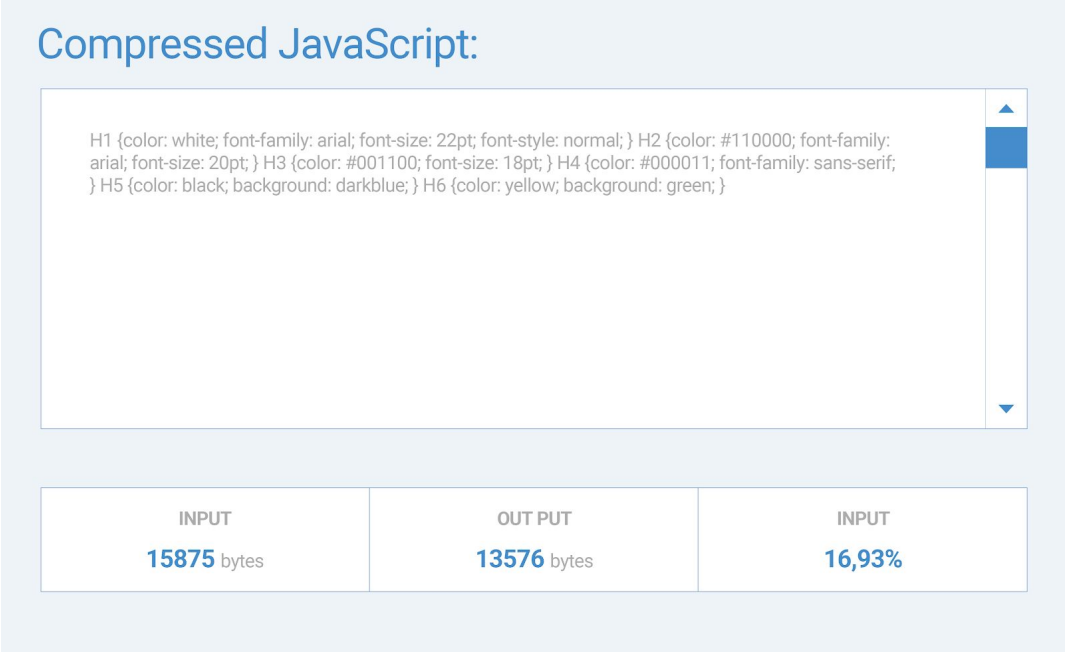
Identify the **css files** that needs to be minified through PSI/GT Metrix results.

Step 2

Go to http://csscompressor.com/ and minify your css files and download the minified version of the CSS files.

e.g.:

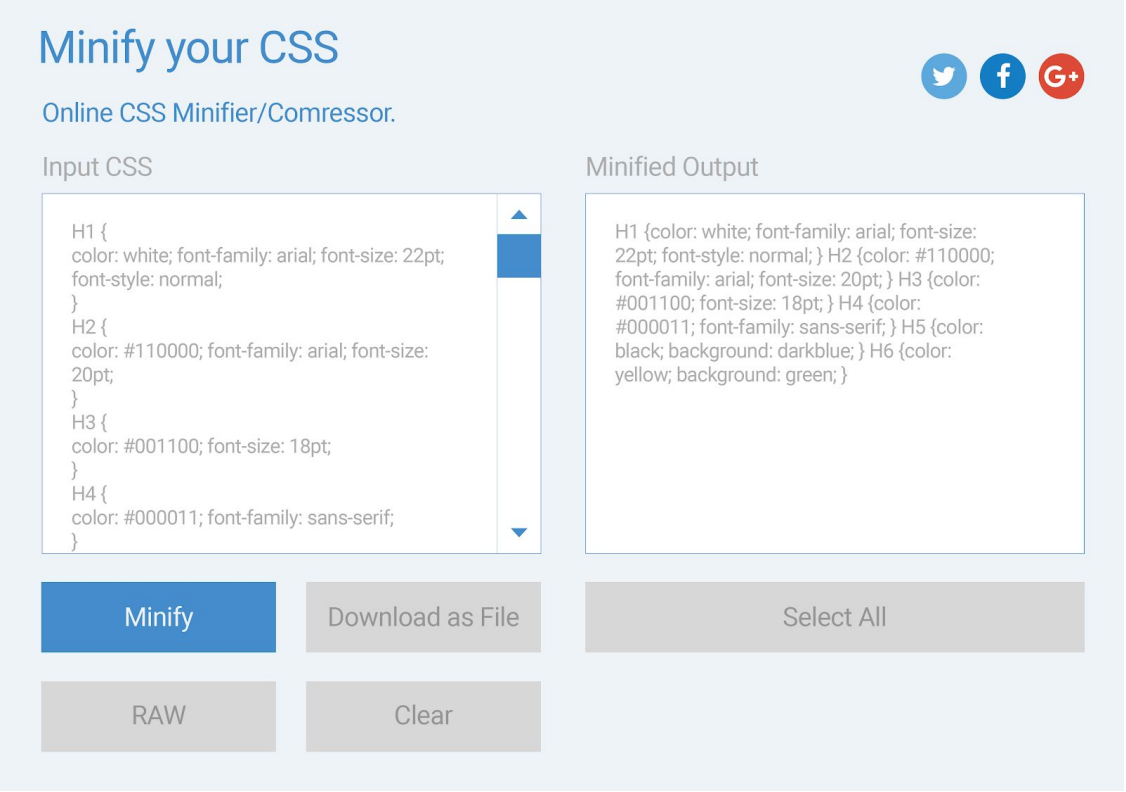
2



Source: internal mockup

Go to the http://www.cssminifier.com and minify your css files and download the minified version of the CSS files.

e.g.:



Source: internal mockup

Step 3

Upload to the live server - Login to CSS folder using FTP / Cpanel details of the site.

Step 4

Locate the existing CSS folder path and back it up internally on your desktop.

Step 5

Upload/replace the old CSS file with optimized CSS file.

Step 6

Perform a round of *Smoke Testing* to ensure there are no negative impact on site.

Step 7

Check the website post changes for improved speed of the mobile website with Page Speed Insight & Web Page Test.

*Note: Minify can be performed by other tools as well (Ex : Notepad++, http://www.minifier.org etc..); however must be used at one's own risk and take precautions. Google will not be responsible for any issues while using these external tools to minify the scripts.

Minify JavaScript

Many times the size of the JS files are huge and we can reduce the size significantly by minifying it, means by removing the unnecessary spaces between codes.

Significance

Minify helps to reduce the size of the JS file results less load on network and faster load time.

Requirement

CPanel / FTP access with elevated privileges will need to be enabled to **add/modify JS files**.

References

Bowdidge, Robert. "Compressing Your JavaScript with Closure Compiler | PageSpeed Insights | Google Developers." Google Developers. Google Inc., *18 Aug. 2015. Web. 03 Nov. 2016.*

High Level Procedure

1. Identify the JS files to minify.
2. Run Javascript file which identified for minification through http://www.jslint.com/ to validate the source code for any common mistakes (Ex - Missing semicolons at the end of a line)
3. Minify the JS files (using any tool of the choice) and replace with original JS files.
4. Save the changes.

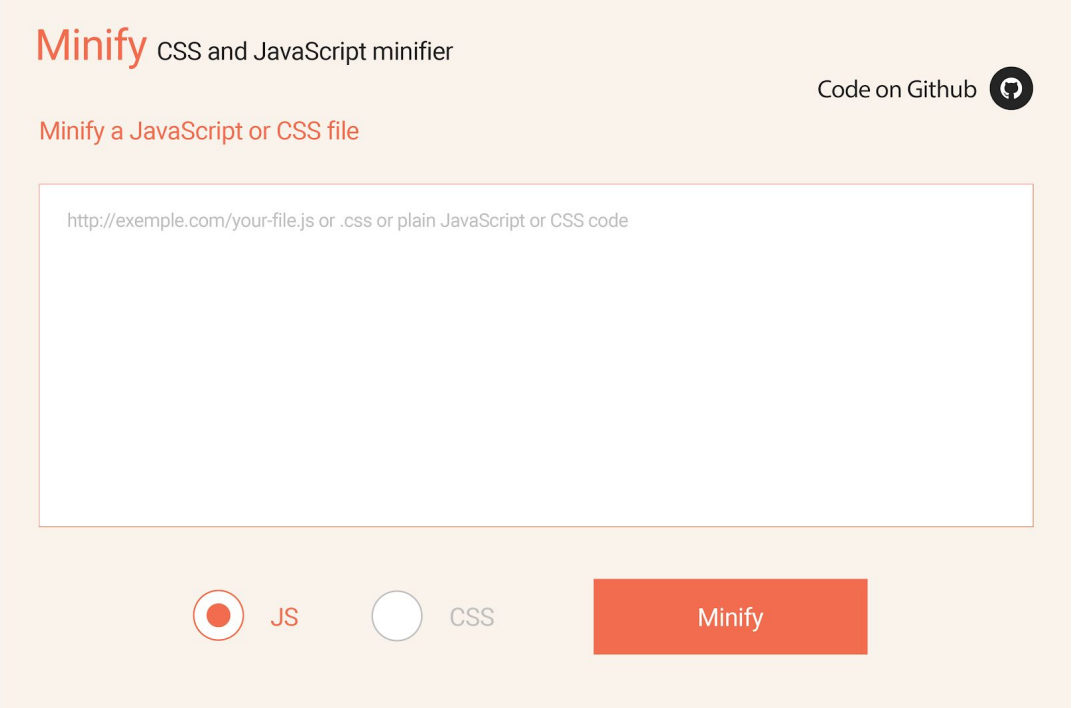
Detail Procedure

Step 1

Identify the *.JS file/s* that can be minified.

Step 2

Go to http://www.minifier.org and minify the **JS** files and then download minified version of the **JS** files.
e.g.:



Source: internal mockup

Step 3

Uploading to live server - Login to server using FTP /CPanel details of the website.

Step 4

Locate the existing JS folder path and take a required backup on server.

Step 5

Upload/replace with the optimized JS file.

Step 6

Perform a round of *Smoke Testing* to ensure there are no negative impact on site.

Step 7

Check the website post changes for improved speed of the mobile website with Page Speed Insight & Web Page Test.

*Note: Minify can be performed by other external tools (e.g.: https://jscompress.com/ etc.), however must be used at one's own risk and take precautions. Google will not be responsible for any issues while using these external tools to minify the scripts.

Eliminate Render-Blocking JavaScript in Above-the-Fold Content

The browser builds the DOM tree before rendering a page by parsing the HTML markup. Whenever the parser encounters a JS script it has to stop and execute it before it can continue parsing the HTML.

Significance

In the case of an external JS script the parser is also forced to wait for the resource to download, which may incur one or more network roundtrips and delay the time to first render of the page. We need to avoid and minimize the use of blocking JavaScript, particularly external JS scripts that must be fetched before they can be executed.

References

Remove Render-Blocking JavaScript

Scope

The team will not perform script ***Inline*** activity directly to customer sites and only recommend the customer to do it if the customer finds it is possible.

High Level Procedure

1. Identify all the JS files which causes render blocking using Page Speed Index tool.
2. Use the ***"defer"*** attribute to defer them, and then place the JS files into the bottom of the HTML page.
3. Save the changes.

Detail Procedure

Step 1

List down all the ***.js*** files causing for ***render blocking javaScripts*** from ***PSI*** (Pagespeed Insight) page.

Step 2

Pick one ***.js*** script and check the size of the java script.

Step 3

If the size is reasonably small then recommend the customer to make the script ***"Inlined"*** **if possible from customer side** that is the contents of the script will be added directly into the HTML of the page.

Step 4

If the size of the script is big & the script is not crucial and can wait to load, then "***defer"*** the script (at bottom of the script) to load and make sure that deferring should not impact the importance of displaying.

e.g.: Source: HTML <script> defer Attribute

```
<script src="demo_defer.js" defer></script>
```

If deferring script breaks the website it's most likely because of inlined script in the page.
Use this workaround to fix the issue : And use ->
https://gist.github.com/Antoinebr/e6e20213d80ec104b64e2b4348c3ce26
This technique is only in case of inline JS

Step 5

Go to the **Step 2** and pick the next ***.js*** script and repeat for all the in scope ***.js*** scripts.

Step 6

Check the website post changes for improved speed of the mobile website with Page Speed Insight & Web Page Test.

Eliminate Render-Blocking CSS in Above-the-Fold Content

The browser builds the DOM (Document Object Model) tree before rendering a page by parsing the HTML markup. Whenever the parser encounters a CSS script it has to stop and execute it before it can continue parsing the HTML.

- Render blocking CSS delays a webpage from being visible in a timely manner.
- Every one of your css files delays your page from rendering.
- The bigger your css, the longer the page takes to load.
- The more css files you have, the longer the page takes to load.

Significance

In the case of an external CSS script the parser is also forced to wait for the resource to download, which may incur one or more network roundtrips and delay the time to first render of the page. We need to avoid and minimize the use of blocking CSS script, particularly external CSS scripts that must be fetched before they can be executed.

References

Render blocking CSS
Optimize CSS Delivery (*Last updated April 26, 2016)*

High Level Procedure

1. Identify all the CSS files which causes render blocking using Page Speed Index tool.
2. Properly label conditional css
3. Lessen the amount of css files in the critical path
    a. Combine CSS
    b. Inline CSS
4. Repeat for all CSS files.
5. Post changes recheck performance of mobile website with Page Speed Insight & Web Page Test.

Detail Procedure

Step 1

Check whether ***.css*** has been loaded before the ***.js*** files. If **"No"** then load the ***.css*** files before the ***.js*** files.

Step 2

List down all the ***.css*** files causing for ***render blocking CSS*** from PSI (Pagespeed Insight) page.

Step 3

Pick one ***.css*** script and check the size of the ***.css*** script.

Step 4

If the size is reasonably small then recommend the customer to make the ***.css*** script ***"Inlined"*** if possible from customer sides. In this case the contents of the .***css*** script will be added directly into the HTML of the page.

Step 5

Check to see if the CSS is labelled correctly for conditional situation.
Examples below:
For printing only ..

```
<link href="print.css" rel="stylesheet" media="print">
```

For use only when page is of certain width.

```
<link href="other.css" rel="stylesheet" media="(min-width: 40em)">
```

Step 6

If there are more than one smaller css files loaded. Each of it consumes additional time to load each additional file called.

Combine the smaller CSS files into one file. This can often shave a second or more off your page load time.

Step 7

Go to the **Step 2** and pick the next ***.css*** script and repeat till the last in scope ***.css*** script.

Step 8

Check the website post changes for improved speed of the mobile website with Page Speed Insight & Web Page Test.

Avoid Landing Page Redirects

Redirections trigger an additional HTTP request-response cycle and delay page rendering. In the best case, each redirect will add a single round trip (HTTP request-response), and in the worst it may result in multiple additional roundtrips to perform the DNS lookup, TCP handshake, and TLS negotiation in addition to the additional HTTP request-response cycle.

e.g.:

- ***example.com*** uses responsive web design, no redirects are needed - fast and optimal!
- ***example.com*** → ***m.example.com/home*** - multi-roundtrip penalty for mobile users.
- ***example.com*** → ***www.example.com*** → ***m.example.com*** - very slow mobile experience.

Significance

As a result, you need to minimize use of redirects to improve site performance.

Scope

Recommend the customer for changing the HTML code. The team will not do it and request the customer only to do it if possible.

References

Avoid Landing Page Redirects (*Last updated April 8, 2015)*

High Level Procedure

1. Look for mobile redirect urls.
2. Look for mobile URLs specified in Adwords account
   Refer this
3. Advice to keep minimum redirects if possible.

Detail Procedure

Step 1

Open Web Page Test.

Step 2

Look for mobile URLs specified in Adwords Account.
Refer to this link for doing this.

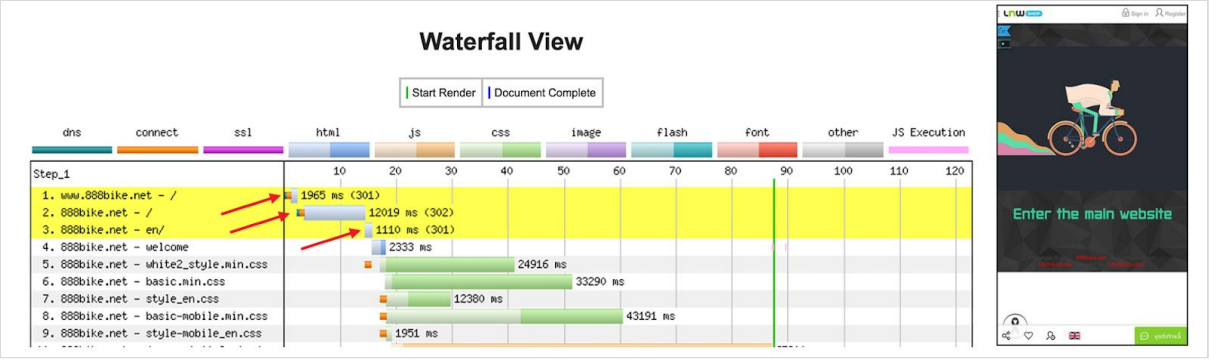
Step 3

Paste the mobile website URL and click Test.

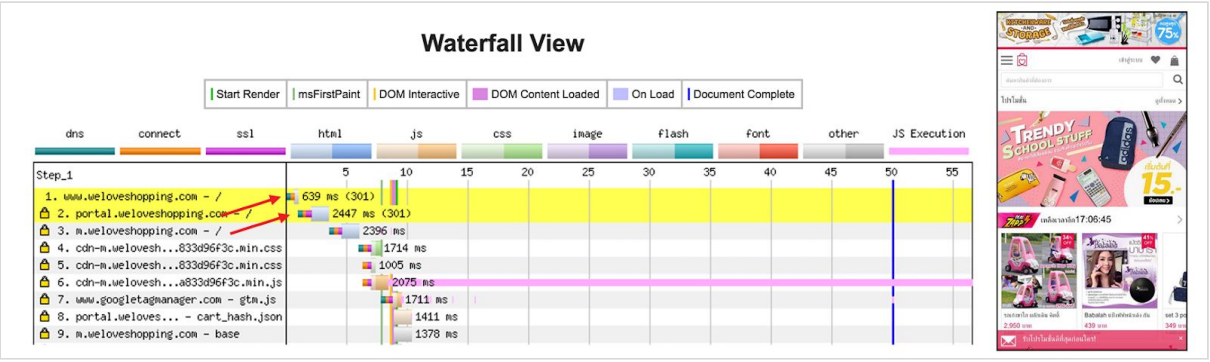
Step 4

Check for redirect urls in ***Waterfall chart*** for the in scope mobile website.
e.g.: Example of Dell & Amazon sites for redirections.

Two Redirects



Single Redirect



Source: Internal testing at Google

Step 4

Advising for changing to minimum redirects if possible.

Step 5

If reduction in redirect is possible, then do a smoke test to see any negative impact post changes.

Step 6

Check the website post changes for improved speed of the mobile website with Page Speed Insight & Web Page Test.

Prioritize Visible Content

Visible content is the portion of a Web Page users see on their screen before they scroll. Sometimes referred to as "above the fold". If the amount of data required to exceeds the initial congestion window (typically 14.6kB compressed), it will require an additional round trips between your server and the user's browser.

Prioritize content that is initially visible to a user: Websites that seem very fast and crisp to load are often just as large as slow websites, they have just prioritized above the fold content so that the site appears to load very fast. Users love webpages that show content quickly and so does Google. This article will provide some guidance on how to concentrate on above the fold content for significantly faster page loads. If you are getting the "prioritize visible content" error on the Google Page Speed Insight tool, this will help you fix it.

Reducing the size of the above the fold content: Google suggests two main strategies for accomplishing this, and they provide us a pretty good outline for examining our pages. We will cover both in depth.

- Structure your HTML to load the critical, above-the-fold content first
- Reduce the amount of data used by your resources

Prioritize main content: One of the quickest, simplest and common ways to improve how a user perceives your webpage load time is to ensure your HTML is presenting the content of your web page first before it is presenting other things.

Significance

For users on networks with high latencies such as mobile networks, this can cause significant delays to page loading.

References

- Reduce the size of the above-the-fold content (*Last updated April 8, 2015)*
- Content Priority Guide

High Level Procedure

1. Look for the ***sidebar*** code sections in the HTML code.
2. Place the ***content*** code sections before the ***sidebar*** code section in the HTML code.
3. Save the changes.

Detail Procedure

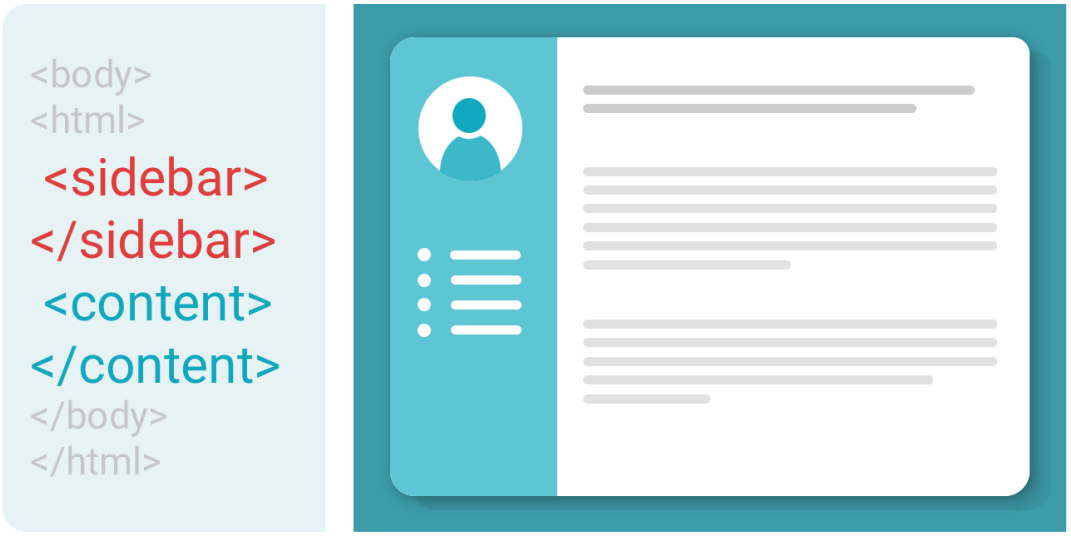
Step 1

Check whether there is any Sidebar in the Web Page.

Step 2

If there is sidebar, ask the customer to go to HTML code and check whether the ***sidebar code section*** is above the ***content area***.

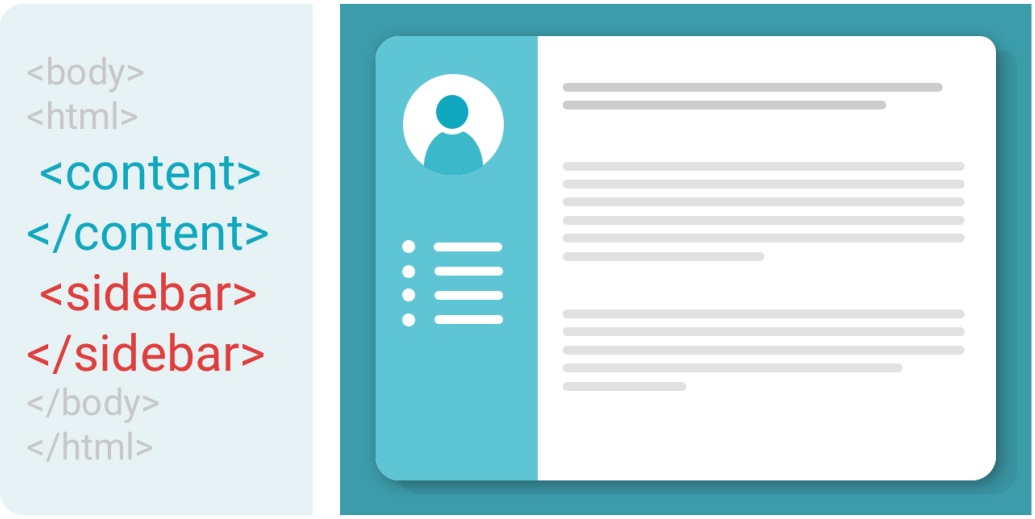
e.g.:



Source: internal mockup

Step 3

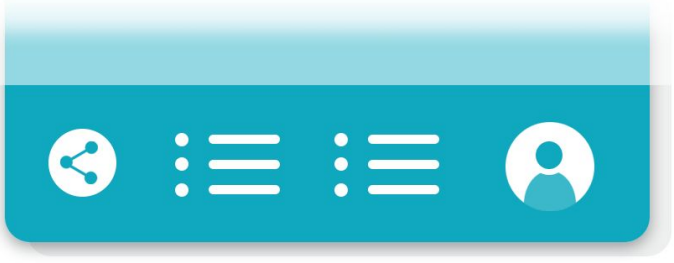
If the ***sidebar code section*** is above the ***content area*** then advise to make the ***sidebar code section*** below the ***content area.***
e.g.:



Source: internal mockup

Step 4

Check for footers, place them at the bottom of the page.
e.g.:



Source: internal mockup

Step 5

Save the changes and refresh the web page.

Step 6

Check the website post changes for improved speed of the mobile website with Page Speed Insight & Web Page Test.

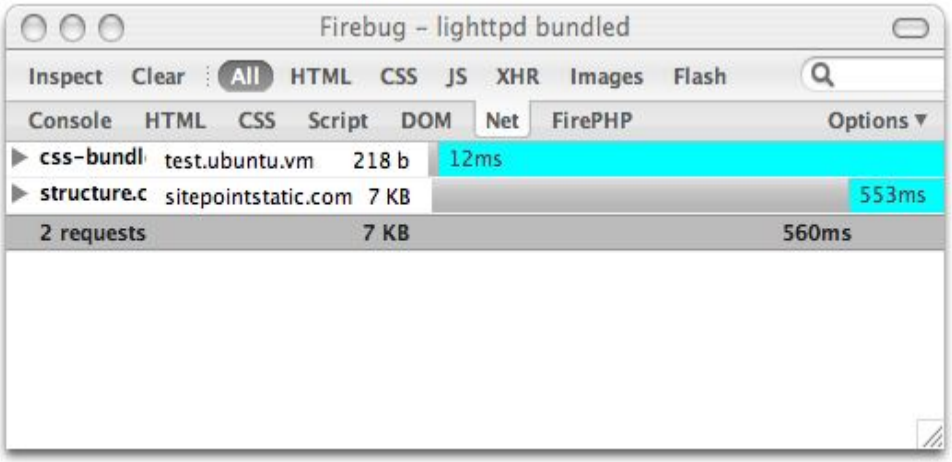
Bundling Static Resources (JS / CSS)

Considerable amount of time is taken when browsers make requests for resources from the server. This time can be reduced if multiple CSS/JS are bundled/combined into a single file. This is because, the browser will have fewer requests to make thereby saving time in receiving the resource. The following table gives an idea of the possible difference that this technique could make in a sample program.

	Combining Resources	Without Combining Resources	Change
File Requests	9	34	256%
KB Sent	3.26	11.92	266%
KB Received	388.51	530	36%
Load Time	510 MS	780 MS	53%

Source: internal mockup

The table below illustrates how additional requests(more than 6) are processed in a browser.



Source: internal testing at Google

Requirements

CPanel / FTP / SSH access with elevated privileges to modify and upload files.

References

- Combine external javascript
- Combine JavaScript

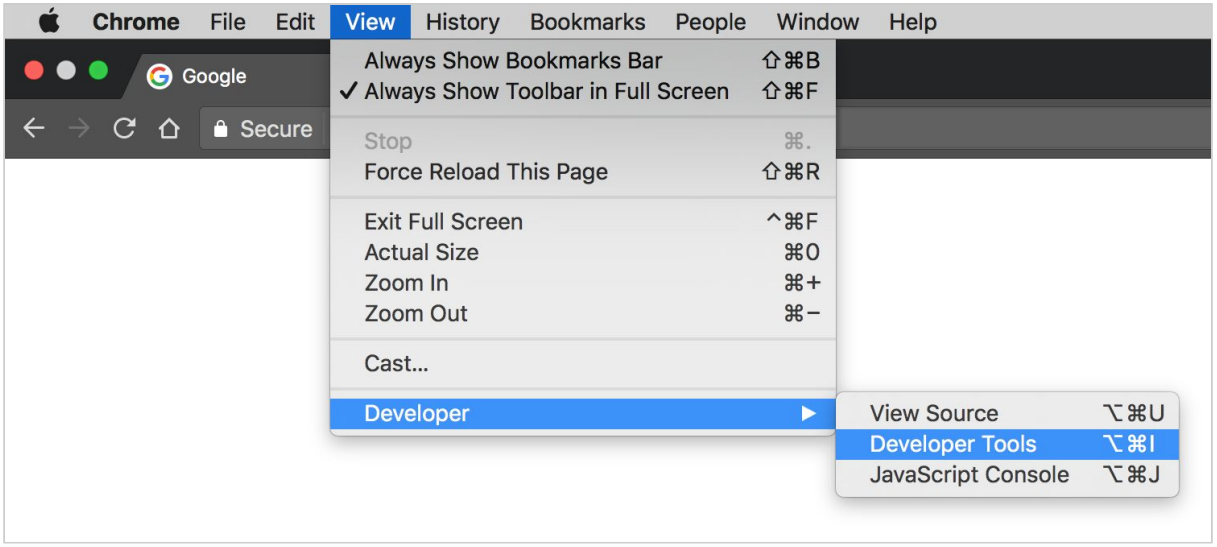
Criteria

Most browsers currently limit the number of simultaneous connections per hostname to six. If there are six or more JS/CSS files in a site, then this technique can be suggested.

Detailed Procedure

Step 1

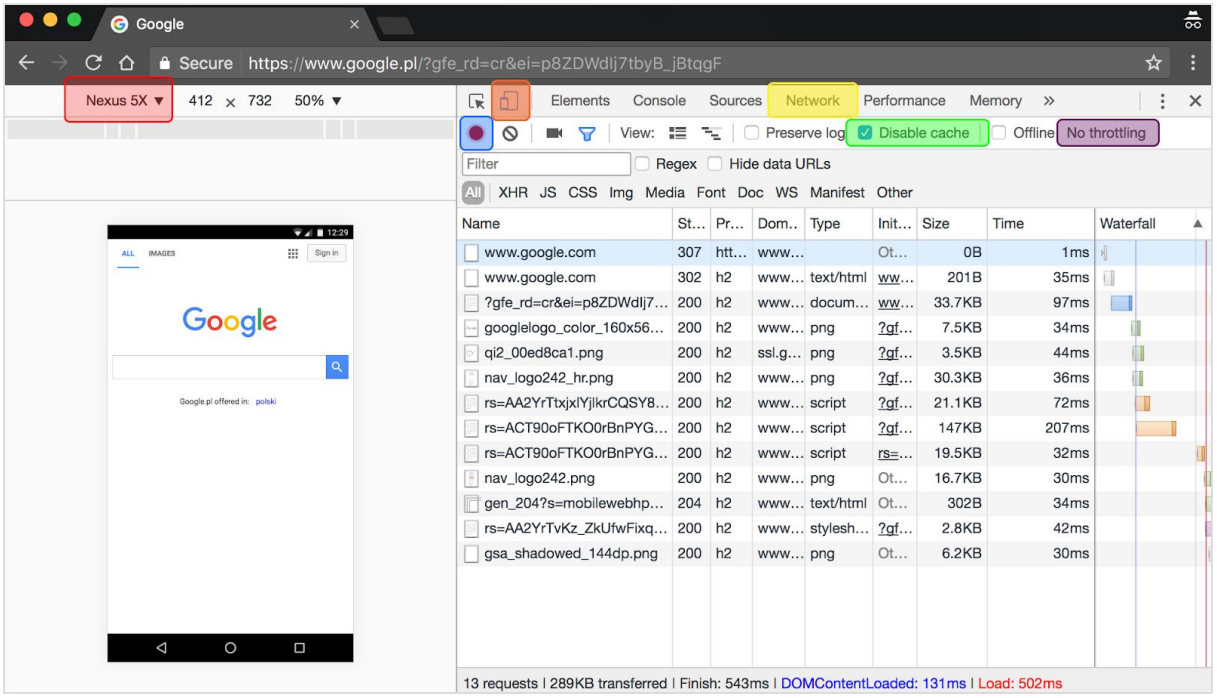
Navigate to Chrome DevTools, Set Up Chrome DevTools, and Record a Page Load of the URL you are testing.



Source: Internal testing at Google

Step 2

Select “JS” to filter items to only show JS files from Network settings.



Source: internal testing at Google

Step 3

Look at the footer of the page where it says X / Y requests. Y is the total number of resources requested.

Step 4

Repeat steps 1-2 with “CSS”. Return back to Chrome DevTools. Look again at the footer of the page where it says X / Y requests (for CSS).

Step 5

Remove all reference links to old JS / CSS files and update with new bundled JS / CSS from app / website.

Step 6

Save and upload the modified file to the server and then refresh the webpage on the browser.

Step 7

Perform a round of *Smoke Testing* to make sure that there are no negative impacts on the website.

Step 8

Check the website post changes for improved speed of the mobile website with PSI / Web Page Test / GT Metrix.

Note: Always take backup of old JS files.

Use of CDN (Content Delivery Network)

This is a group of servers strategically placed across the globe for helping in delivering static content to end users much faster. CDN will be useful when a website has users in different countries and when web pages include a lot of static resources such as images, videos, audio clips, CSS files, and JavaScript.

Pros of CDN

- Faster content delivery
- Less load on the origin
- Increased reliability (Handle traffic spikes and reduce system overload)
- Reduce bandwidth cost

Possible Cons of CDN

- Blocked access
- Additional Cost

Significance

Use of a CDN can provide a faster way to fetch the static contents and improve the web page load time. CDN also can dramatically decrease the latency. If maximum percentage of server response time is due to static resources, then we can recommend adopting CDN service.

References

What Is a CDN?

High Level Procedure

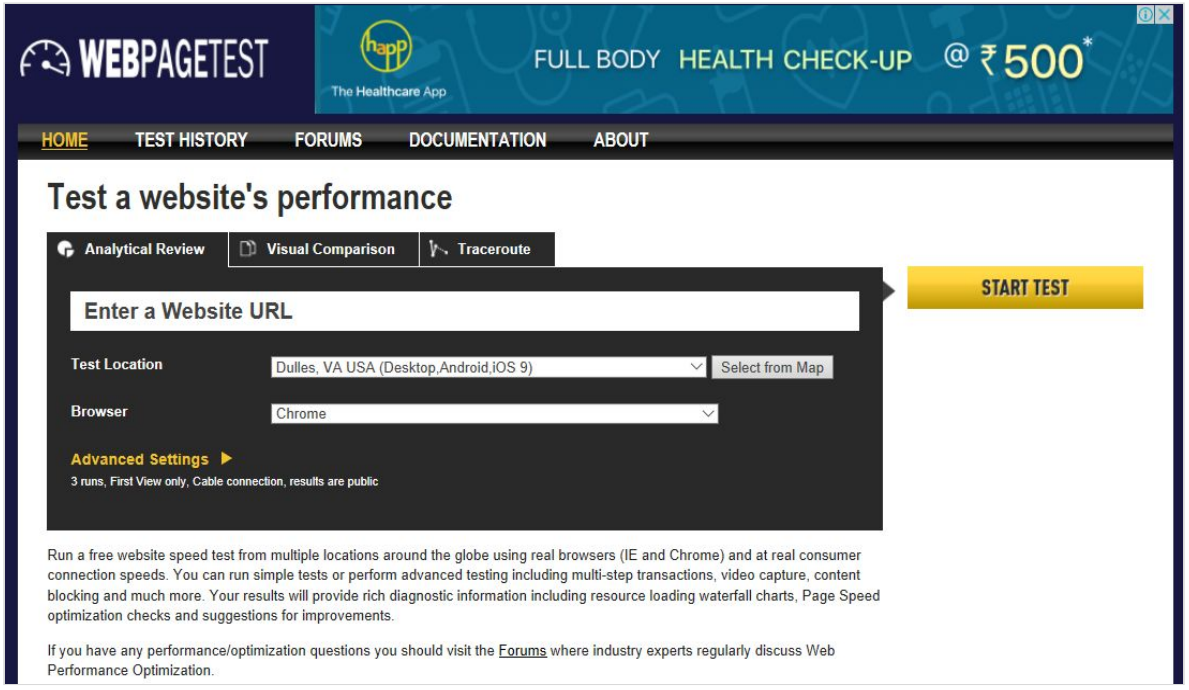
1. Check for the use of CDN

2. If it is not used, then recommending to use the CDN technology.

Detail Procedure

Step 1

Go to http://www.webpagetest.org/

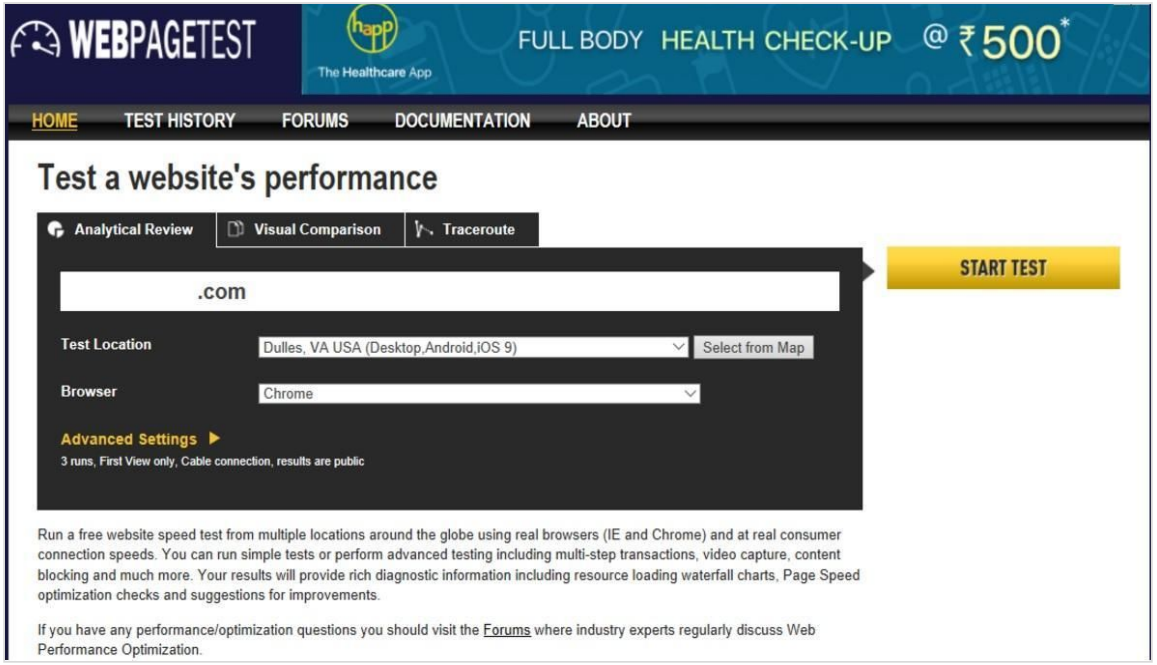


Step 2

Make sure that ***"Analytical Review"*** is selected.

Step 3

Type/past the mobile website URL into the portal.



Step 4

Select the ***"Test Location"*.**

Step 5

Select the ***"Browser"*.**

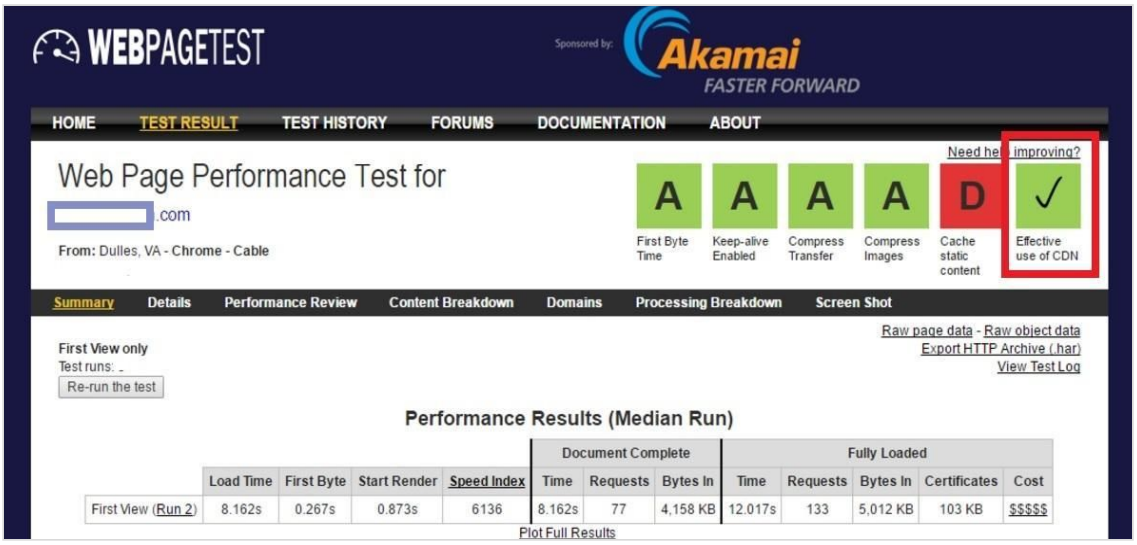
Step 6

Click ***"Start Test"***.

Step 7

Once the test is completed, look for the ***"Effelctive use of CDN"*** in upper right corner in the page. If ***"Effective use of CDN"*** is **RED/crossed** **(X)** that means CDN has not been used properly for all the static contents. Click the cross **(X)** to see the effective score and the list of static assets for which CDN can be used.

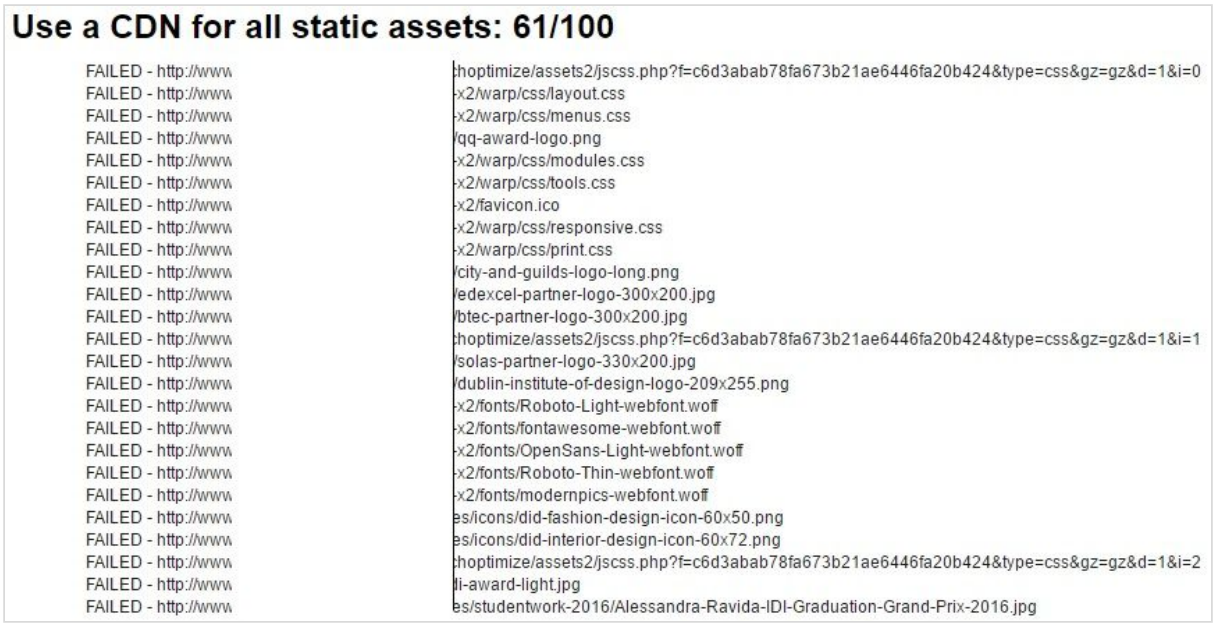
e.g.: In below example, CDN has been used optimally.



Source: Internal Experiment at Google & http://www.webpagetest.org/

e.g.: In below example, CDN has not been used optimally.

e.g.: Score for use of CDN and the list of static assets for which CDN can be used.



Use a CDN for all static assets: 61/100

FAILED - http://www	choptimize/assets2/jscss.php?f=c6d3abab78fa673b21ae6446fa20b424&type=css&gz=gz&d=1&i=0
FAILED - http://www	-x2/warp/css/layout.css
FAILED - http://www	-x2/warp/css/menus.css
FAILED - http://www	/qq-award-logo.png
FAILED - http://www	-x2/warp/css/modules.css
FAILED - http://www	-x2/warp/css/tools.css
FAILED - http://www	-x2/favicon.ico
FAILED - http://www	-x2/warp/css/responsive.css
FAILED - http://www	-x2/warp/css/print.css
FAILED - http://www	/city-and-guilds-logo-long.png
FAILED - http://www	/edexcel-partner-logo-300x200.jpg
FAILED - http://www	/btec-partner-logo-300x200.jpg
FAILED - http://www	choptimize/assets2/jscss.php?f=c6d3abab78fa673b21ae6446fa20b424&type=css&gz=gz&d=1&i=1
FAILED - http://www	/solas-partner-logo-330x200.jpg
FAILED - http://www	/dublin-institute-of-design-logo-209x255.png
FAILED - http://www	-x2/fonts/Roboto-Light-webfont.woff
FAILED - http://www	-x2/fonts/fontawesome-webfont.woff
FAILED - http://www	-x2/fonts/OpenSans-Light-webfont.woff
FAILED - http://www	-x2/fonts/Roboto-Thin-webfont.woff
FAILED - http://www	-x2/fonts/modernpics-webfont.woff
FAILED - http://www	es/icons/did-fashion-design-icon-60x50.png
FAILED - http://www	es/icons/did-interior-design-icon-60x72.png
FAILED - http://www	choptimize/assets2/jscss.php?f=c6d3abab78fa673b21ae6446fa20b424&type=css&gz=gz&d=1&i=2
FAILED - http://www	li-award-light.jpg
FAILED - http://www	es/studentwork-2016/Alessandra-Ravida-IDI-Graduation-Grand-Prix-2016.jpg

Step 8

Recommending to use CDN to improve the load time of the webpage.

Step 9

Check the website post changes for improved speed of the mobile website with Page Speed Insight & Web Page Test.

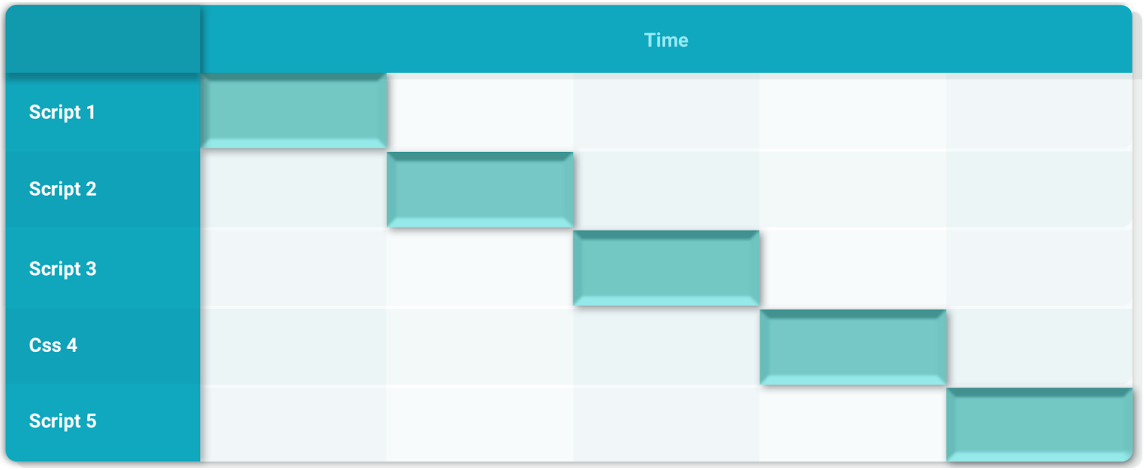
Enable Asynchronous Scripts

Normally browsers do not display content until all external resources are fully loaded. This is called “synchronous loading”. However, you could override this functionality to make certain elements load without waiting for all external resources to load. This is known as "asynchronous loading".

Using asynchronous scripts ensures that your page renders more quickly since users are not forced to wait for a script to finish downloading before the page renders.

The diagrams given below should explain this concept better:

When all resources load **synchronously**, the elements will load in order, like this:



Source: internal mockup

When resources load **asynchronously**, it will load simultaneously with other scripts, like this:

Source: internal mockup

You can see in the diagram above, that **scripts 2 and 3** are now able to load at the same time, which speeds up the overall loading of a page.

Requirements

- CPanel / FTP / SSH access with elevated privileges to modify / upload files.

References

- Synchronous and Asynchronous Loading
- Use Asynchronous Scripts

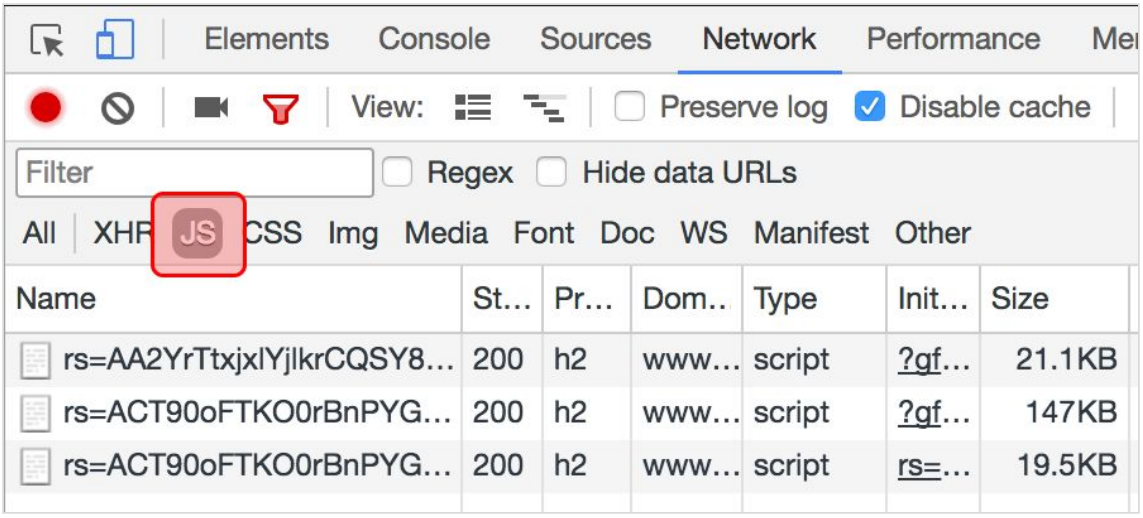
Detailed Procedure

Step 1

Navigate to Chrome DevTools, set Up Chrome DevTools, and Record a Page Load of the URL you are testing.

Step 2

Select “JS” to filter items to only show JS files from network settings tab.



Source: internal testing at Google

Step 3

Analyze the JS resources that loads in the waterfall and identify the resources which need to be loaded Asynchronously.

Step 4

Update async attribute to the resources. Here's what a page's <head> tag would look like when all resources are loaded in asynchronous manner:

```
<html>
    <head>
        <script async src="2.js" />
          </head>
    <body>Hello World!</body>
</html>
```

Step 5

Save and upload the modified file to server and then refresh the webpage on the browser.

Step 6

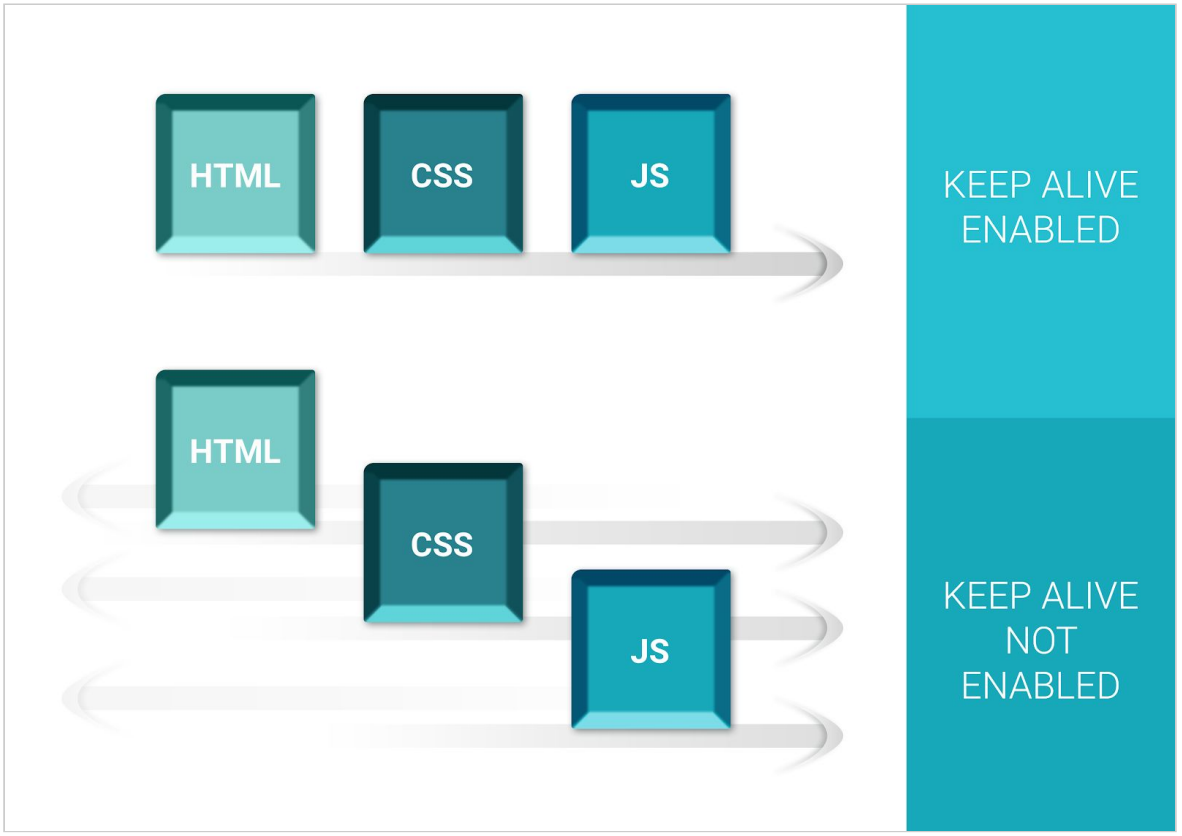
Perform a round of *Smoke Testing* to make sure there are zero negative impacts on the website.

Step 7

Check the website post changes for improved speed of the mobile website with Web Page Test / GT Metrix.

Enable Keep-Alive

Enabling keep-alive tells the browser of your visitors to establish a TCP connection only once instead of establishing the connection multiple times to request all the web files loaded on your web page (like images, CSS scripts, Javascripts, etc..) thereby saving time in establishing multiple connections.



Source: internal mockup

Steps to identify if “keep alive” is enabled and enable keep alive if necessary

Step 1

Run website through https://www.giftofspeed.com/check-keep-alive/ HTTP Header of your URL and tell you whether keep-alive connections are enabled by detecting the

keep-alive configurations listed in the header response.

Step 2

Enable Keep-alive through .htaccess / Apache / IIS

Requirements

CPanel / FTP / RDP / SSH access for root user with elevated privileges to modify the web.config / .htaccess and config files

References

- How to enable keep alive
- Enable the HTTP Keep-Alive Response Header

Detailed Procedure

Enable Keep-Alive for Apache servers

Apache enables Keep-Alive connections by default, however you can explicitly turn them on by adding the following line to your httpd.conf file

Step 1

Edit this file /etc/httpd/conf/httpd.conf

Step 2

Then run the following command: sudo vim /etc/httpd/conf/httpd.conf

Step 3

Now add this in the file (Any line should do the trick): KeepAlive On
When you've updated httpd.conf, you'll see a screen like this:

```
#
# KeepAlive: Whether or not to allow persistent connections (more than
# one request per connection). Set to "Off" to deactivate.
#
KeepAlive On

#
# MaxKeepAliveRequests: The maximum number of requests to allow
# during a persistent connection. Set to 0 to allow an unlimited amount.
# We recommend you leave this number high, for maximum performance.
#
MaxKeepAliveRequests 100

#
# KeepAliveTimeout: Number of seconds to wait for the next request from the
# same client on the same connection.
#
KeepAliveTimeout 100
```

Step 4

In order for the new settings to take effect you will need to restart your httpd service by performing the following command: sudo service httpd restart

Enable keep-alive using .htaccess

To enable keep-alive through .htaccess you need to add the following code to your .htaccess file:

```
<ifModule mod_headers.c>
Header set Connection keep-alive
</ifModule>
```

Adding this to your .htaccess file will add keep alive headers to your requests, which will override most web server or host limitations.

Enabling Keep-Alive for NGINX

Keep alive issues can be tackled using the HttpCoreModule. There is a specific directive you should look out for... "keepalive_disable". If you see this make sure you know why it is disabling keep-alive before removing.

Enabling Keep-Alive for IIS

You can perform this procedure by using the user interface (UI), by running Appcmd.exe commands in a command-line window, by editing configuration files directly, or by writing WMI scripts.

User Interface

Step 1

Open Internet Information Services (IIS) Manager

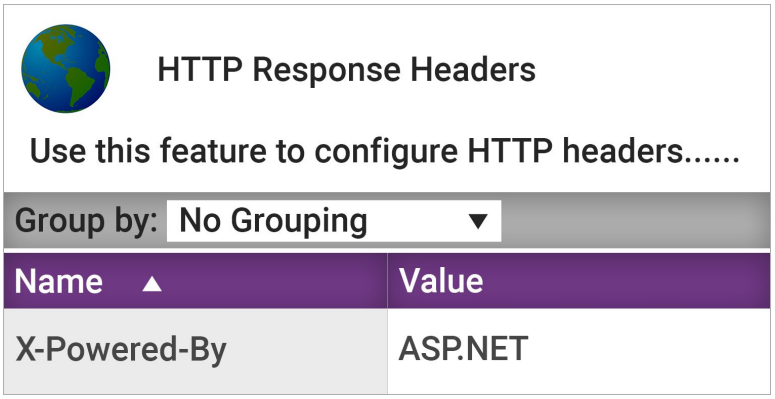
Step 2

In Features View, double-click HTTP Response Headers

Source: internal mockup

Step 3

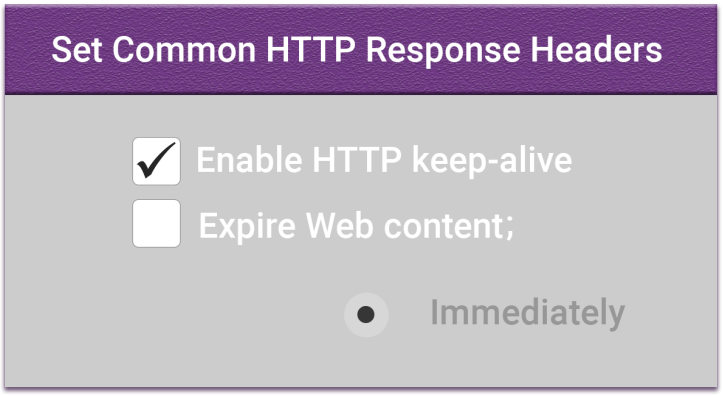
On the HTTP Response Headers page, in the Actions pane, click Set Common Headers.



Source: internal mockup

Step 4

In the Set Common HTTP Response Headers dialog box, check the box to enable HTTP keep-alives, and then click OK.



Source: internal mockup

Command Line

To enable the HTTP keep-alive header, use the following syntax:

appcmd set config /section:httpProtocol /allowKeepAlive:true

Use CSS Sprites

A web page with many images, particularly many small images, such as icons, buttons, etc. can take a long time to load and generates multiple server requests. Using the image sprites instead of separate images will significantly reduce the number of HTTP requests a browser makes to the server, which can be very effective for improving the loading time of web pages and overall site performance.

Requirements

CPanel / FTP / RDP access for root user with elevated privileges to modify the files

References

- CSS Sprites with Background Positioning
- CSS Sprites

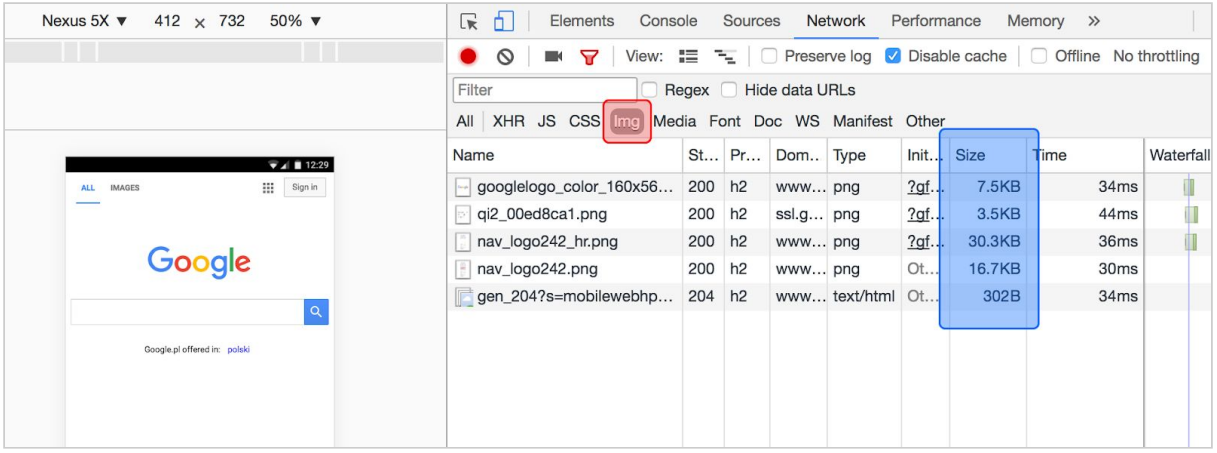
Steps to Identify if CSS sprites are used and include to sprites if required

Step 1

Navigate to Chrome DevTools, Set Up Chrome DevTools, and Record a Page Load of the URL you are testing.

Step 2

Select “Images” to filter items to only show files from Network settings.



Source: internal testing at Google

Step 3

Analyze the images listed on “Img Tab” and make a list of images which can be updated on sprite sheet (Ex : Icon, Thumbnails etc..)

Step 4

Combine images into CSS sprites - Below is an example to combine images

Let's say we have two images we want to display on a webpage for style purposes and wish to combine them into one. We must know the size of the images in order to create the sprite. We will use an Example where both images are the same size (50 pixels by 50 pixels).

Source: internal mockup

Step 5

For this example will we use the images as background images in divs. This means that we will create empty div tags in our HTML to display images. If we wanted the megaphone image to show up somewhere on our page we could create a CSS div class of "megaphone” and same for “Smile”.

```
.megaphone {width:50px; height:50px; background:url(images/sprite.png) 0
0px;}

.smile {width:50px; height:50px; background:url(images/sprite.png) 0 -50px;}
```

Step 6

Where we want a megaphone / smile, just enter an empty div called “megaphone” or “smile”.

```
<div class="megaphone"></div>
<div class="smile"></div>
```

Avoid CSS @Import

It is a method of importing one css file from inside another css file. This method of retrieving CSS files creates some problems that affect your page speed. The largest problem is that it causes files to load sequentially (one has to wait for the other) instead of in parallel (at the same time). This wastes time and round trips and makes your web page load slower.

Requirement

CPanel / FTP / RDP access for root user with elevated privileges to modify the files

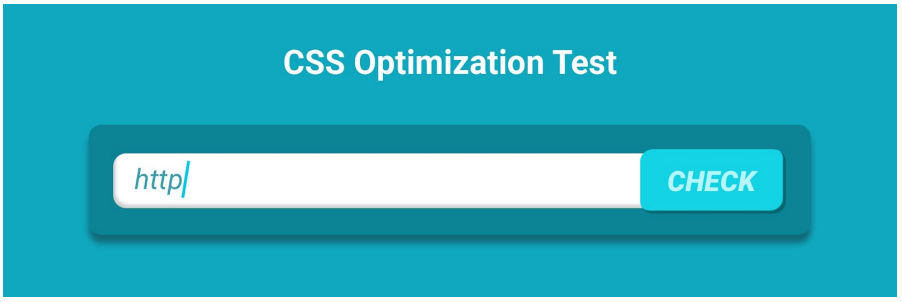
References

- @import
- Avoid CSS @Import

Detailed Procedure

Step 1

Use this CSS tool to find out if you have @imports on your page



Source: internal mockup

Step 2

List down the CSS files which are using @ CSS Import

Here is an example of a Import call:

```
@import url("style.css")
@import url("style-auto.css")
```

Step 3

Remove all the @import rules from your CSS code. Then open all the CSS files you use for the web page (also the ones you don't use @import for), copy & paste all the codes within those files, combine them all in one large CSS script and call it from a HTML header within style tags (the same can be done with Javascripts). Combining all CSS scripts in one bigger script and inlining it in the HTML markup is the best way to deal with CSS scripts when it comes to page speed. There is no reason to call separate CSS files (or even a single one), unless your site uses extremely large CSS scripts.

```
<link rel="stylesheet" href="style.css" type="text/css">
<link rel="stylesheet" href="style-auto.css" type="text/css">
```

Appendix

Tools Used

The following tools can be used for Mobile Site performance optimizations.

Sr. No.	Tools	Descriptions	URL
1.	Buitwith	This tool displays the information about the technologies used by the mobile web page.	www.BuiltWith.com
2.	Wappalyzer	This tool shows the technology stacks used to build the website. We need to install the plugin to use this tool.	https://wappalyzer.com/
3.	CMS Detector	This tool shows the CMS used in the website.	http://guess.scritch.org/
4.	Web Server Information	This tool shows the web servers used to build the website.	http://browserspy.dk/webserver.php
5.	Page Speed Insight	This tool gives a Page Speed Insight score which ranges from 0 to 100 based on how well a webpage adheres to best practices (better the score, better the speed!), and also a list of specific recommendations which help to speed-optimize the webpage.	https://developers.google.com/speed/pagespeed/insights/

6.	Web Page Test	This tool gives the load time of the web page, and relevant information like component wise loading time, snapshots of each page navigations etc.	http://www.webpagetest.org/
7.	GTmetrix	GTmetrix gives us the insight on how well your site loads and provides actionable recommendations on how to optimize it.	https://gtmetrix.com/
8.	CSS Minifier	Minify the CSS files.	http://www.cssminifier.com
9.	Minifier	Minify the both JS & CSS files.	http://www.minifier.org

Treatment List

Treatment ID	Treatment Name
T1	Enabling Browser Caching
T2	Enable Compression
T3	Avoiding Loading Duplicate JS & CSS Files
T4	Image Optimization - Removing Duplicate Images
T5	Image Optimization - Compress Images
T6	Minify CSS
T7	Minify JavaScript
T8	Eliminate Render-Blocking JavaScript in Above-the-Fold Content
T9	Eliminate Render-Blocking CSS in Above-the-Fold Content
T10	Avoid Landing Page Redirects
T11	Prioritize Visible Content
T12	Use of CDN (Content Delivery Network)
T13	Bunding Static Resources (JS/CSS)
T14	Enable Asynchronous Scripts
T15	Enable Keep Alive
T16	Use CSS Sprites
T17	Avoid CSS@Import

<Additional Treatments- To be removed>

Image Optimization - Issue with Background Images

This problem commonly occurs in one of two ways:

- ➢ Your responsive site loads images sized for both desktop and mobile. Of course, only one is shown.
- ➢ Your responsive site hides images that are shown on the desktop site, but they're still loaded.

Significance

If you hide an image with CSS, the browser still loads it. Loading unused resources unnecessarily slows page load. To re-check this action item, use: ***ChromeDevTool > Network > All > Filter by Name.***

References

- Optimize Images
- Image Optimization *(Ilya Grigorik, Last updated February 9, 2017)*
- How to Build Responsive Images with srcset

High Level Procedure

1. Open Page Speed Insight (PSI) and run the website and download the optimized images from Page Speed Insight. Analyse the background image loading.
2. Open chrome Dev Tool and go to ***Network > All > Filter by Name*** and look for if any additional background images need to be optimized.

3. Recommending to use ***"srcset"*** attribute for image loading into the website.

Detail Procedure

Step 1

Use Page Speed Insight to get the images detailed required for optimization.

Step 2

Explain the developer about the issue that the images can be loaded in an optimal way as per the required size based on specific devices. Some of the big images are getting downloaded which are big for mobile devices and increases the image loading time which results overall high response time.

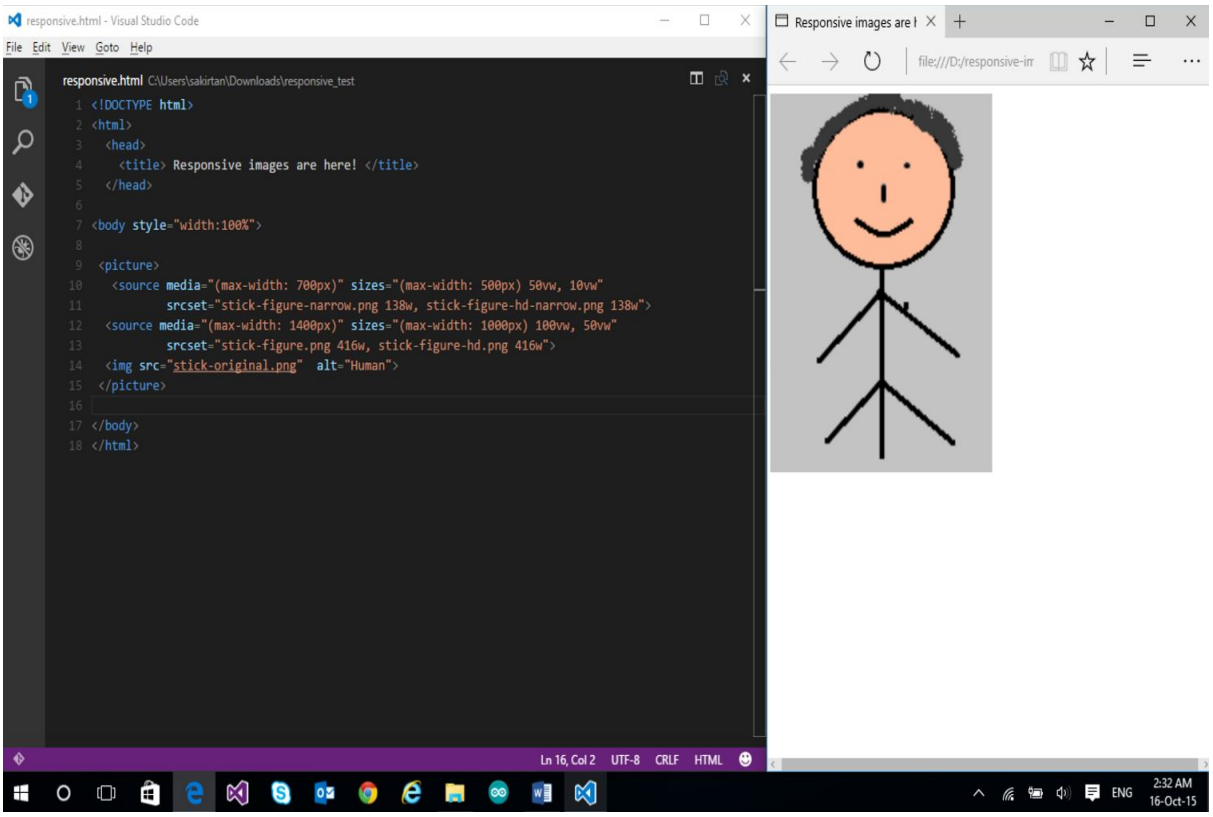
Step 3

Recommending to use ***"srcset"*** attribute for image loading into the website.

- ➢ The "***srcset"*** attribute allows you to provide multiple image files for different devices. The browser will choose the best image for the device, for example using a **2x** image on a **2x** display, and only the most appropriate image is downloaded and displayed.

- ➢ On browsers that don't support "***srcset"***, the browser simply uses the ***default*** image file specified by the "***src"*** attribute. That is why it is important to always include a **1x** image that can be displayed on any device.

e.g.:



Source: https://www.sitepoint.com/how-to-build-responsive-images-with-srcset/

Case 1

When we want to display separate images based on the device-pixel ratio, we'd go with basic *"srcset"* implementation as below:

```
<img src="images/space-needle.jpg"
srcset="images/space-needle.jpg 1x, images/space-needle-2x.jpg 2x,
images/space-needle-hd.jpg 3x">
```

Case 2

When we want a different size (height, width) image on a larger or smaller viewport:

```
<img src="images/space-needle.jpg"
srcset="images/space-needle.jpg 200w, images/space-needle-2x.jpg 400w,
images/space-needle-hd.jpg 600w">
```

Case 3

For Background Images use the Technique shown in following example.

HTML

```
<div class="my-bg"></div>
```

* *

CSS

```
.my-bg{
  background-image: url(http://lorempicsum.com/futurama/1200/600/3);
  height: 600px;
  width: 1200px;
  max-width: 100%;
  background-repeat: no-repeat;
  display: block;
  margin: auto;

}

@media only screen and (max-width: 768px){
  .my-bg{
     background-image: none;
  }
}
```

Step 4

Verify the improvement of the mobile web page speed with Page Speed Insight and Web Page Test.

Enabling Caching Module for Apache

There are a few cases in which some of the servers do not have the caching and compression modules enabled. For those cases we can follow the below mentioned approach to enable the modules. However, we do not recommend doing this manually as this could lead to negative issues on the site. The advertiser can reach out to the server providers and they will enable it and it's an easy fix.

Significance

Without enabling this module, the caching mechanism will not work.

High Level Procedure

1. Request the Web Server Administrator to check whether the Caching module is enabled or not.
2. If the module is not enabled then enable it.

Detail Procedure

Step 1

Request the Web Server Administrator to check whether the Caching module is enabled or not.

Step 2

If the module is not enabled, then enable it.

Step 3

Request them to share the details of the modules.

Step 4

Check the website post changes for improved speed of the mobile website with Page Speed Insight & Web Page Test.

Enabling Caching Module for IIS

There are a few cases in which some of the servers do not have the caching and compression modules enabled. For those cases we can follow the below mentioned approach to enable the modules. However, we do not recommend doing this manually as this could lead to negative issues on the site. The advertiser can reach out to the server providers and they will enable it and it's an easy fix.

Significance

Without enabling this module, the caching mechanism will not work.

References

Leverage Browser Caching *(Last updated April 8, 2015)*

High Level Procedure

1. Request the Web Server Administrator to check whether the Caching module is enabled or not.
2. If the module is not enabled then enable it.

Detail Procedure

Step 1

Request the Web Server Administrator to check whether the Caching module is enabled or not.

Step 2

If the module is not enabled then enable it.

Step 3

Request them to share the details of the modules.

Step 4

Check the website post changes for improved speed of the mobile website with Page Speed Insight & Web Page Test.

Enabling Caching Module for Nginx

There are a few cases in which some of the servers do not have the caching and compression modules enabled. For those cases we can follow the below mentioned approach to enable the modules. However, we do not recommend doing this manually as this could lead to negative issues on the site. The advertiser can reach out to the server providers and they will enable it and it's an easy fix.

Significance

Without enabling this module, the caching mechanism will not work.

References

Leverage Browser Caching *(Last updated April 8, 2015)*

High Level Procedure

1. Request the Web Server Administrator to check whether the Caching module is enabled or not.
2. If the module is not enabled then enable it.

Detail Procedure

Step 1

Request the Web Server Administrator to check whether the Caching module is enabled or not.

Step 2

If the module is not enabled then enable it.

Step 3

Request them to share the details of the modules.

Step 4

Check the website post changes for improved speed of the mobile website with Page Speed Insight & Web Page Test.

Enable Compression Module for Apache

There are a few cases in which some of the servers do not have the compression modules enabled. For those cases we can follow the below mentioned approach to enable the module. However, we do not recommend doing this manually as this could lead to negative issues on the site. The advertiser can reach out to the server providers and they will enable it and it's an easy fix.

Significance

Without enabling this module, the compression mechanism will not work.

References

Enable Compression (*Last updated April 8, 2015*)

High Level Procedure

1. Request the Web Server Administrator to check whether the Compression module is enabled or not.
2. If the module is not enabled then enable it.

Detail Procedure

Step 1

Request the Web Server Administrator to check whether the Compression module is enabled or not.

Step 2

If the module is not enabled then enable it.

Step 3

Request them to share the details of the modules.

Step 4

Check the website post changes for improved speed of the mobile website with Page Speed Insight & Web Page Test.

Enable Compression Module for IIS

There are a few cases in which some of the servers do not have the compression modules enabled. For those cases we can follow the below mentioned approach to enable the modules. However, we do not recommend doing this manually as this could lead to negative issues on the site. The advertiser can reach out to the server providers and they will enable it and it's an easy fix.

Significance

Without enabling this module, the compression mechanism will not work.

References

Enable Compression (*Last updated April 8, 2015*)

High Level Procedure

1. Request the Web Server Administrator to check whether the Compression module is enabled or not.
2. If the module is not enabled then enable it.

Detail Procedure

Step 1

Request the Web Server Administrator to check whether the Compression module is enabled or not.

Step 2

If the module is not enabled then enable it.

Step 3

Request them to share the details of the module.

Step 4

Check the website post changes for improved speed of the mobile website with Page Speed Insight & Web Page Test.

Enable Compression Module for Nginx

There are a few cases in which some of the servers do not have the compression modules enabled. For those cases we can follow the below mentioned approach to enable the modules. However, we do not recommend doing this manually as this could lead to negative issues on the site. The advertiser can reach out to the server providers and they will enable it and it's an easy fix.

Significance

Without enabling this module, the compression mechanism will not work.

References

Enable Compression (*Last updated April 8, 2015*)

High Level Procedure

1. Request the Web Server Administrator to check whether the Compression module is enabled or not.
2. If the module is not enabled then enable it.

Detail Procedure

Step 1

Request the Web Server Administrator to check whether the Compression module is enabled or not.

Step 2

If the module is not enabled then enable it.

Step 3

Request them to share the details of the modules.

Step 4

Check the website post changes for improved speed of the mobile website with Page Speed Insight & Web Page Test.

HTML Minification

Many times the size of the HTML page is huge and we can reduce the size significantly by minifying it, means by removing the unnecessary spaces between codes.

Significance

Minify helps to reduce the size of the HTML page results less load on network and faster load time.

Requirement

CPanel / FTP access with elevated privileges will need to be enabled to **add/modify HTML page**.

References

- Minify Resources (*Google Developers. Google Inc., 26 Apr. 2016)*
- HTML Minifier
- HTML Minifier

High Level Procedure

1. Identify the HTML script to minify.
2. Minify the HTML script (using any tool of the choice) and replace with original HTML script.
3. Save the changes.

Detail Procedure

Step 1

Identify the *.JS file/s* that can be minified.

Step 2

Go to http://kangax.github.io/html-minifier or any other your prefered tool, and minify the **HTML page** and then download minified version of the **HTML code**.
e.g.: http://kangax.github.io/html-minifier Web site view

Source: http://kangax.github.io/html-minifier/

Step 3

Uploading to live server - Login to server using FTP /CPanel details of the website.

Step 4

Locate the existing HTML code folder path and take a required backup on server.

Step 5

Upload/replace with the optimized HTML code.

Step 6

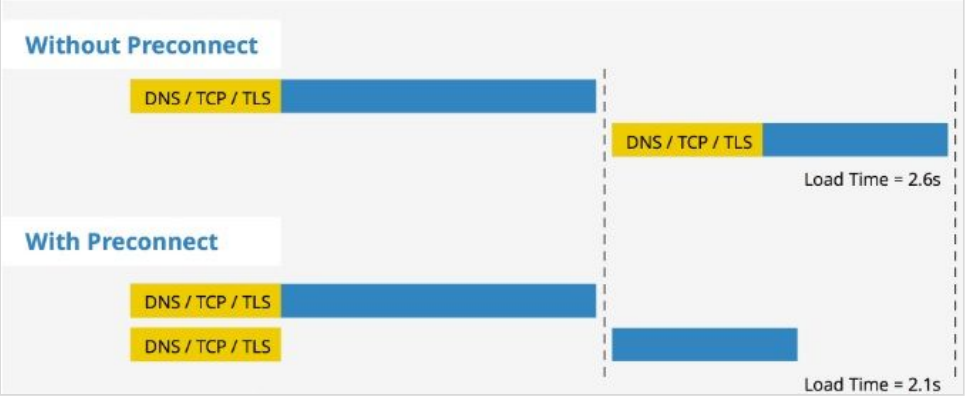
Perform a round of *Smoke Testing* to ensure there are no negative impact on site.

Step 7

Check the website post changes for improved speed of the mobile website with Page Speed Insight & Web Page Test.

Enable Preconnect

Before an HTTP request is actually sent to the server, preconnect technique allows the browser to set up the early connections. The connections like TCP Handshake, DNS Lookup, and TLS negotiation can be initiated beforehand that results in eliminating roundtrip latency for these connections.



Source: https://www.keycdn.com/support/preconnect/

Significance

Improve overall round trip latency of the mobile webpage.

References

- Initiate a preconnect
- Preconnect

High Level Procedure

1. Resolve the URL defined by the href attribute.
2. Let origin be preconnect URL's origin.
3. Let corsAttributeState be the current state of the element's crossorigin content attribute.
4. Let credentials be a boolean value set to true.
5. If corsAttributeState is Anonymous and origin is not equal to current Document's origin, set credentials to false.
6. Attempt to obtain connection with origin and credentials.
7. Save the changes.

Detail Procedure

According to W3C Resource Hints, we need to follow the following steps (Step 1 to Step 6) to initialize the preconnect.

Step 1

Resolve the URL defined by the href attribute.

Step 2

Let origin be preconnect URL's origin.

Step 3

Let corsAttributeState be the current state of the element's crossorigin content attribute.

Step 4

Let credentials be a boolean value set to true.

Step 5

If corsAttributeState is Anonymous and origin is not equal to current Document's origin, set credentials to false.

Step 6

Attempt to obtain connection with origin and credentials.

Step 7

Perform a round of *Smoke Testing* to ensure there are no negative impact on the site.

Step 8

Check the website post changes for improved speed of the mobile website with Page Speed Insight & Web Page Test.

Enable Lazy Loading Images

Lazy loading is when you stop a web object (usually images or scripts) from loading until the visitor of your website actually needs to load them. i.e. when you lazy load images, your images only load once a user scrolls down the page. This makes your initial page load time much faster, as you only have to load the additional resources if a reader engages with your page.Since, on an average, images make up 63% of a webpage's size lazy loading images will ensure that the user has the best possible experience on the website.

Requirements

CPanel / FTP / SSH access with elevated privileges to modify / upload files.

References

- How to lazy load images
- Lazy Loading Images

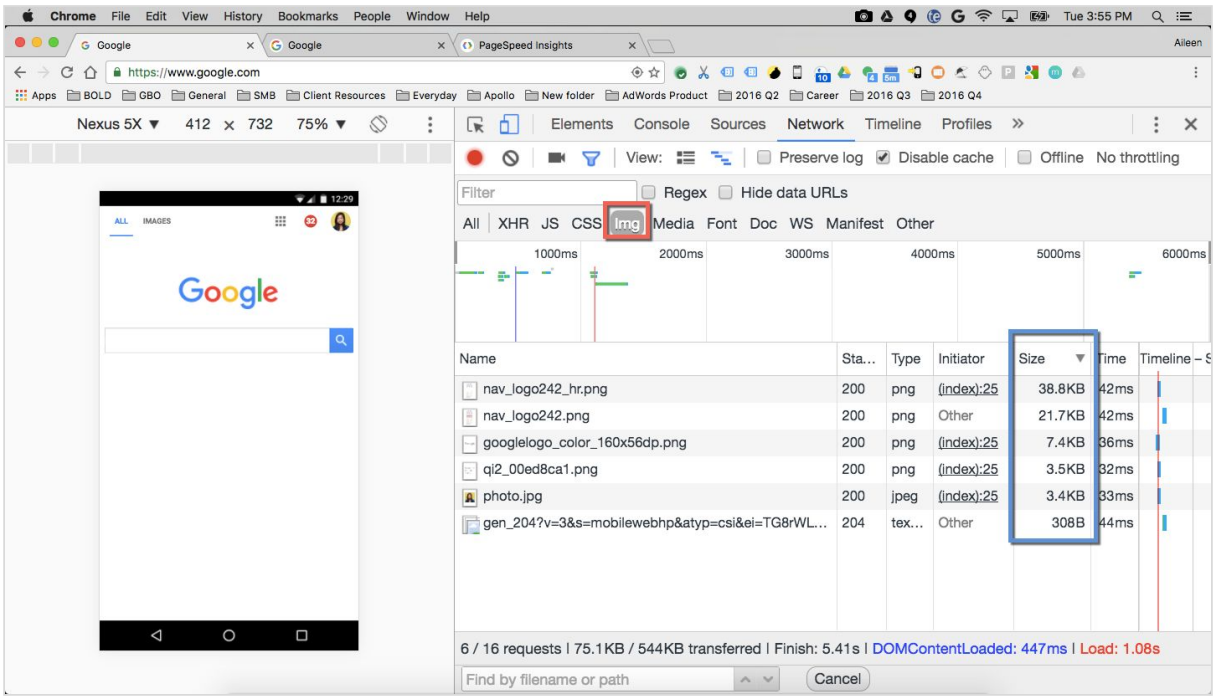
Detailed Procedure

Step 1

Navigate to Chrome DevTools, Set Up Chrome DevTools, and Record a Page Load of the URL you are testing

Step 2

Select "Images" to filter items to only show Image files from Network settings



Step 3

Analyze the images listed on “Img Tab” and make a list of images which require the lazy load script.

Step 4

Lazy Loading scripts - Apply the lazy loading script for the chosen images. There are some good small lazy load scripts available which do not use jQuery, do the job perfectly and are quite easy to set up. Below are examples of scripts that are easily available.

a. Be Lazy.js
b. Echo.JS

You will need to refer to the Blazy or Echo documentation to setup the lazy loading script on the webpage.

Step 5

Perform a round of Smoke Testing to make sure there is no negative impact on the website post updating the website with Lazy Load scripts. The site can also be checked with GT Metrix to gauge if the site is loading faster.

Enable Google Page Speed Module

One of the more recent popular module for Apache is mod_pagespeed. It is an output filter for Apache 2.2+ that can be configured through a variety of options through configuration files or a .htaccess file. An “output filter” is a something that transforms the data before it is sent to the client. In other words, it’s a layer between your website and what the user’s browser receives when they visit your URL.

The goal of mod_pagespeed is to speed up your website. It does this by applying filters to a variety of files in order to reduce the number of trips the browser has to make to grab what it needs, to reduce the size of those files and to optimize the length those files that are cached.

Requirements

SSH access setup for root or a sudo user.

References

- Install on your webserver
- How To Add ngx_pagespeed to Nginx on Debian 8

Detailed Procedure of Installing mod_pagespeed with Apache on an Ubuntu and Debian Cloud Server

Step 1

Install the following packages

- 64-bit version: wget https://dl-ssl.google.com/dl/linux/direct/mod-pagespeed-stable_current_amd64.deb
- 32-bit version: wget https://dl-ssl.google.com/dl/linux/direct/mod-pagespeed-stable_current_i386.deb

Step 2

Make sure the list of packages available via apt-get has been updated using below command

- sudo apt-get update

Step 3

Run the following command:

- sudo dpkg -i mod-pagespeed-*.deb
- apt-get -f install

If the packages need to be installed on CentOS/Fedora run the following command:

- sudo yum install at # if you do not already have 'at' installed
- sudo rpm -U mod-pagespeed-*.rpm

Step 5

Restart the server using $sudo service apache2 restart

Detailed Procedure of Installing ngx_pagespeed packages for Nginx on Ubuntu 14.04, CentOS 7 version or Debian 8 version

Step 1

Create a folder using the command line given below

- **sudo mkdir -p /var/ngx_pagespeed_cache**

Step 2

Make sure to change the ownership of this folder to the Nginx user so that the web server can store files in it:

- **sudo chown -R www-data:www-data /var/ngx_pagespeed_cache**

Step 3

Open the main Nginx configuration file nginx.conf in your favorite text editor

- **sudo nano /etc/nginx/nginx.conf**

Step 4

In this file add the following lines to the http block and save the changes:

```
##
# Pagespeed Settings
##

 pagespeed on;
 pagespeed FileCachePath /var/ngx_pagespeed_cache;</html>
```

Step 5

Add pagespeed configuration lines to every server block file located in

/etc/nginx/sites-available. For example, edit the /etc/nginx/sites-available/default file:

sudo nano /etc/nginx/sites-available/default

Step 6

Add the following lines to the end of the server block:

```
#  Ensure requests for pagespeed optimized resources go to the pagespeed
#  handler and no extraneous headers get set.
location ~ "\.pagespeed\.([a-z]\.)?[a-z]{2}\.[^.]{10}\.[^.]+" { add_header ""
""; }
location ~ "^/ngx_pagespeed_static/" { }
location ~ "^/ngx_pagespeed_beacon" { }
```

The above pagespeed configuration lines ensures that the site's resources are optimized

Step 7

Restart Nginx server for the changes to take effect using the line given below

sudo service nginx restart

Here are some more useful PageSpeed filters:

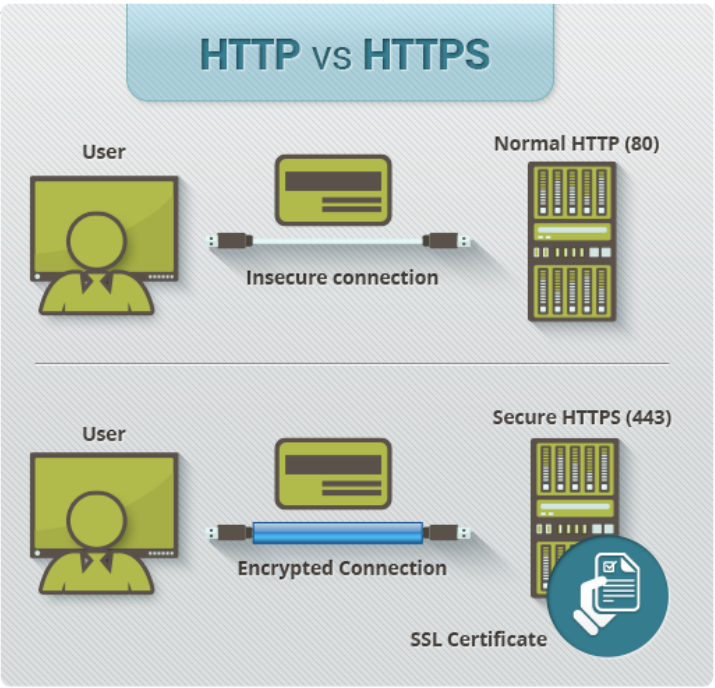
- Canonicalize JavaScript Libraries
- Extend Cache
- Local Storage Cache
- Outline CSS
- Outline JavaScript
- Combine CSS
- Flatten CSS @imports
- Inline CSS
- Inline Google Fonts API CSS
- Combine JavaScript
- Inline JavaScript
- Move CSS Above Scripts

- Configuration file directive to shard domains
- Sprite Images
- Pre-Resolve DNS
- Convert Meta Tags
- Defer Javascript
- Hint Resource Preloading
- Inline Preview Images
- Lazily Load Images

For more details, visit PageSpeed Filters

Enabling HTTPs

HTTPS helps prevent intruders from tampering with the communications between your websites and your users' browsers. Intruders include intentionally malicious attackers, and legitimate but intrusive companies, such as ISPs or hotels that inject ads into pages.



Intruders exploit unprotected communications to trick your users into giving up sensitive information or installing malware, or to insert their own advertisements into your resources. For example, some third parties inject advertisements into websites that potentially break user experiences and create security vulnerabilities.

Detailed Procedure

- Host with a dedicated IP address
- Buy a certificate
- Activate the certificate
- Install the certificate
- Update your site to use HTTPS

Step 1

Host with a dedicated IP address

In order to provide the best security, SSL certificates require your website to have its own dedicated IP address. Lots of smaller web hosting plans put you on a shared IP where multiple other websites are using the same location. With a dedicated IP, you ensure that the traffic going to that IP address is only going to your website and no one else's.

Note: If you don't have a plan with a dedicated IP you can ask your current web host to upgrade your account to have a dedicated IP address.

Step 2

Buy a Certificate

Next you'll need something that proves your website is your website – kind of like an ID Card for your site. This is accomplished by creating an SSL certificate. A certificate is simply a paragraph of letters and numbers that only your site knows, like a really long password. When people visit your site via HTTPS that password is checked, and if it matches, it automatically verifies that your website is who you say it is – and it encrypts everything flowing to and from it.

Technically this is something you can create yourself (called a 'self-signed cert'), but all popular browsers check with "Certificate Authorities" (CA's) which also have a copy of that long password and can vouch for you. In order to be recognized by these authorities, you must purchase a certificate through them.

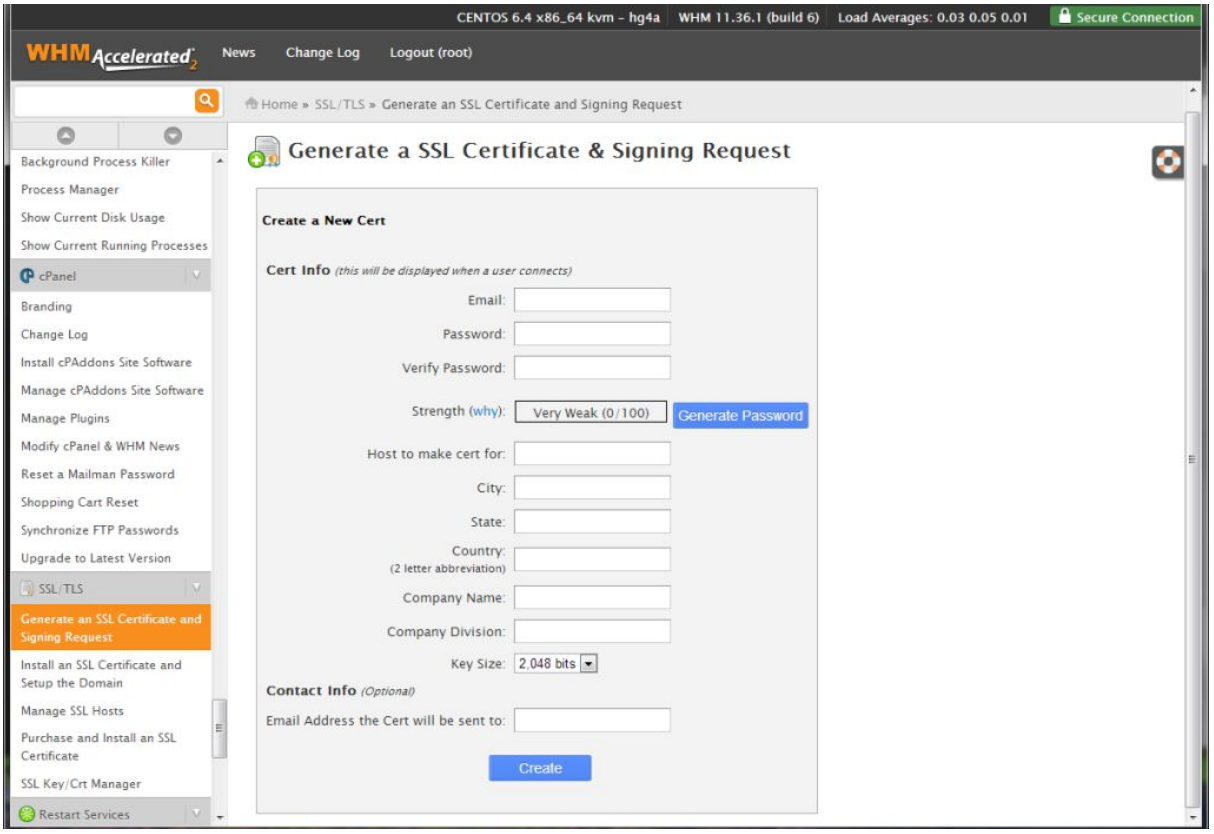
Step 3

Activate the certificate

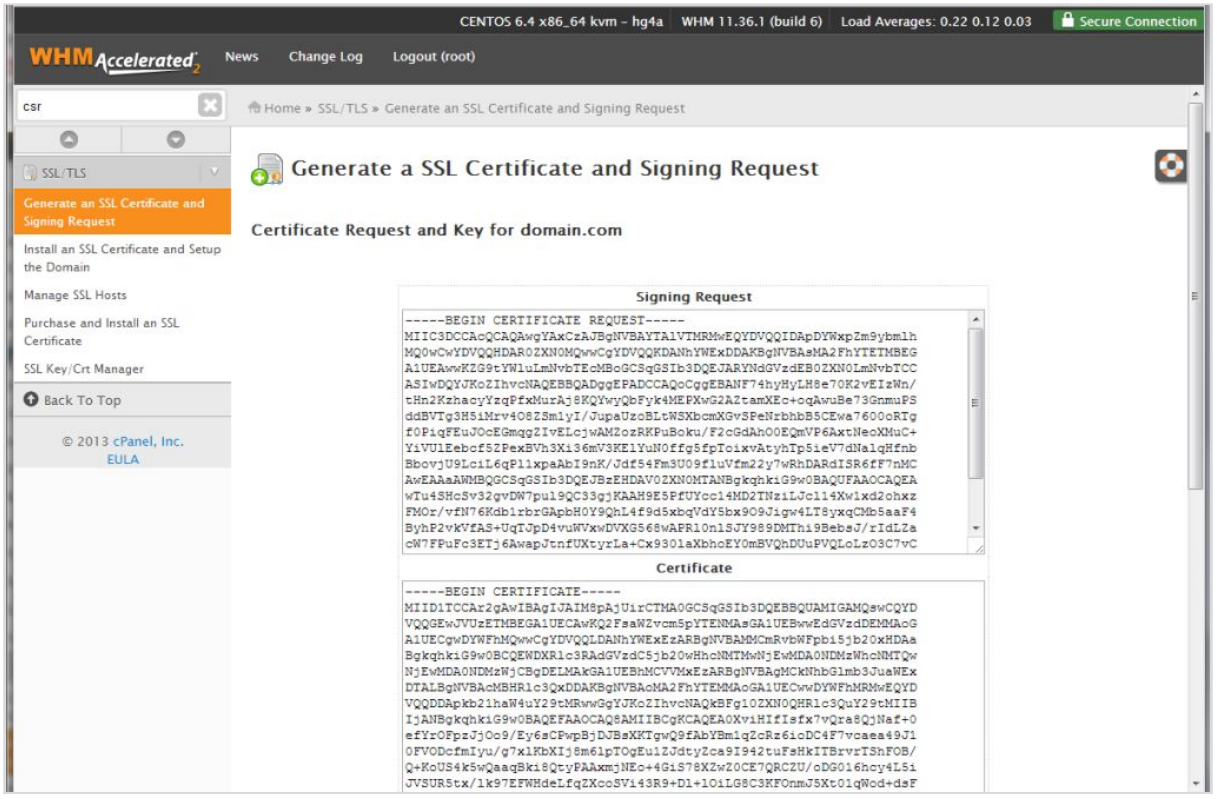
Note: Your web host may do this step for you – check with them before proceeding. This can get complicated and if you can wait 1-2 days it may be best to let them do it.

If you're activating the certificate yourself, the next step is to generate a CSR. It's easiest to do this within your web hosting control panel – such as WHM or cPanel. Go to the SSL/TLS admin area and choose to "Generate an SSL certificate and

Signing Request". Fill out the fields in the screen below:



“Host to make cert for” is your domain name, and the contact email can be blank. When you’ve filled it out, you’ll see a screen like this:

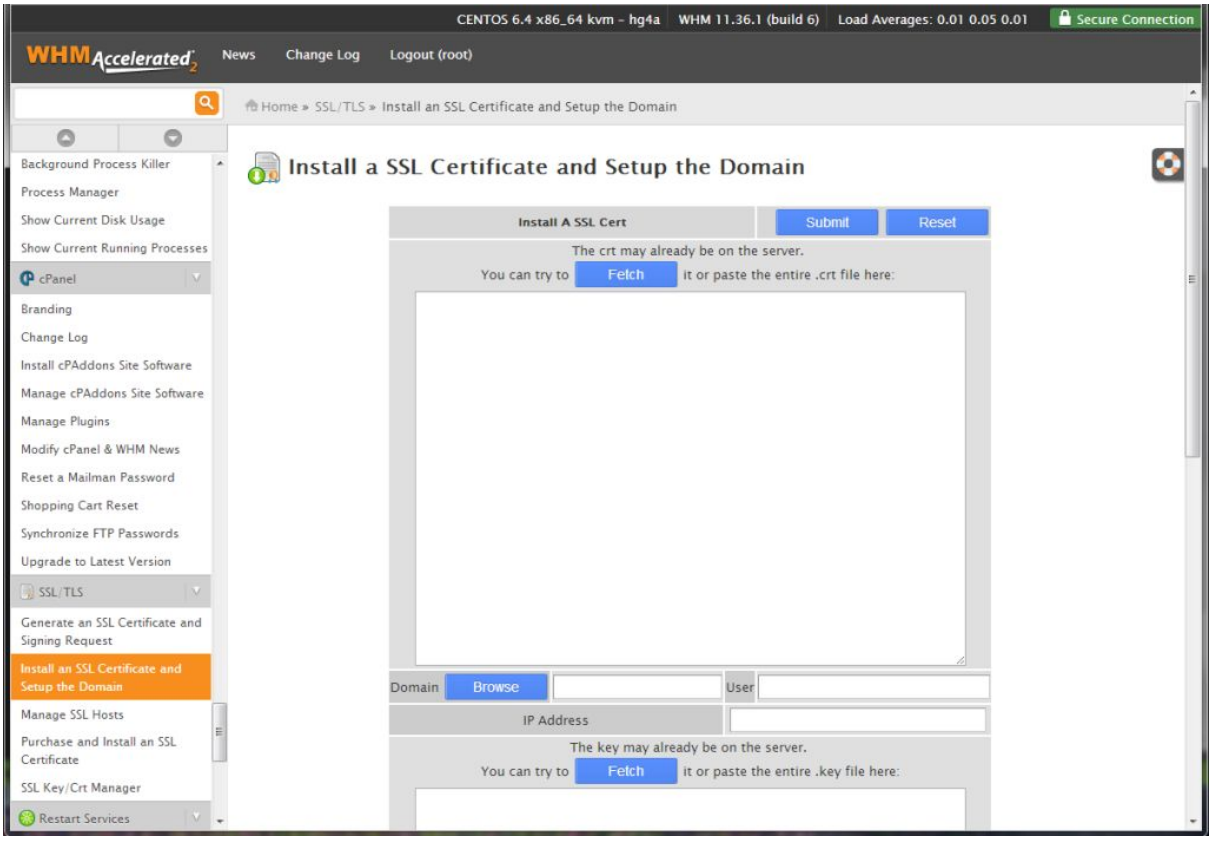


Copy the first block of text. You’ll need this “CSR” to give to the SSL cert issuer so they can establish your identity. Login to your account wherever you bought your certificate and activate it. Paste your CSR and any other fields needed. It will ask you for an approver email. This is an email address that proves you own the domain, ie webmaster@domain.com. If it doesn’t exist, you’ll need to create it so you can get the email that contains the final certificate. Follow the steps and when you are done that email address should have received the cert as a .crt file.

Step 4

Install the certificate

If you're installing up the certificate yourself, this is the easiest step you'll ever do. You have the certificate in hand, all you need to do is paste it into your web host control panel. If you're using WHM.CPanel, click the "Install an SSL Certificate" from under the SSL/TLS menu.



Step 5

Update your site to use HTTPS

At this point if you go to https://yoursite.com you should see it load! Congrats, you've successfully installed SSL and enabled the HTTPS protocol! But your visitors aren't protected just yet, you need to make sure they're accessing your site through HTTPS!

Keep in mind that you typically only need to protect a few pages, such as your login or cart checkout. If you enable HTTPS on pages where the user isn't submitting

sensitive data on there, it's just wasting encryption processing and slowing down the experience. Identify the target pages and perform one of the two methods below.

You can update all links to the target pages to use the HTTPS links. In other words, if there's a link to your cart on your home page, update that link to use the secure link. Do this for all links on all pages pointing to the sensitive URLs.

However, if you want to ensure that people can only use specific pages securely no matter what links they come from, it's best to use a server-side approach to redirect the user if it's not HTTPS. You can do that with a code snippet inserted on top of your secure page. Here's one in PHP:

```
// Require https
if ($_SERVER['HTTPS'] != "on") {
   $url = "https://". $_SERVER['SERVER_NAME'] . $_SERVER['REQUEST_URI'];
   header("Location: $url");
   exit;
}
```

Another server-side approach is to use mod-rewrite. This won't require you to change any of your website files, but will need you to modify your apache configuration.

```
RewriteEngine On
RewriteCond %{HTTPS} off
RewriteRule ^(cart/|checkout/) https://%{HTTP_HOST}%{REQUEST_URI}
```

Dedicated Hosting

Certificate Types

Ensure that you choose a CA that offers the certificate type that you require. Many CAs offer variations of these certificate types under a variety of, often confusing, names and pricing structures. Here is a short description of each type:

- **Single Domain**: Used for a single domain, e.g. example.com. Note that additional subdomains, such as www.example.com, are not included
- **Wildcard**: Used for a domain and any of its subdomains. For example, a wildcard certificate for *.example.com can also be used for www.example.com and store.example.com

- **Multiple Domain**: Known as a SAN or UC certificate, these can be used with multiple domains and subdomains that are added to the Subject Alternative Name field. For example, a single multi-domain certificate could be used with example.com, www.example.com and example.net

Apache: Certificate Signing Request (CSR) Generation Instructions

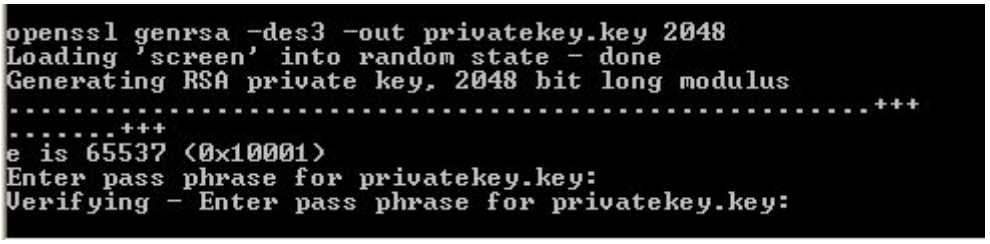
Step 1

Generate Private Key

The OpenSSL package and is usually installed under /usr/local/ssl/bin.
If the utility was installed elsewhere, these instructions will need to be adjusted accordingly.

Type the following command at the prompt:

openssl genrsa -des3 -out <private key file name>.key 2048



NOTE: All certificates that will expire after October 2013 must have a 2048 bit key size.

NOTE: If using Apache on Windows or generating a private key for use with Amazon Web Service, Plesk or cPanel please do not include -des3 as this option is not supported.

NOTE: To bypass the pass phrase requirement, omit the -des3 option when generating the private key. If the private key is left unprotected,SSL Provider recommends access to the server be restricted so that only authorized server

administrators can access or read the private key file.

Step 2

Certificate Signing Request (CSR)

Type the following command at the prompt:

openssl req -new -key <private key file name>.key -out <csr file name>.csr

```
openssl req -new -key privatekey.key -out request.csr
Enter pass phrase for privatekey.key:_
```

NOTE: If using openSSL on Windows, the path to openssl.cnf may need to be specified, such as the following:

openssl req -new -key <private key file name>.key -config "c:\Apache Software Foundation\Apache2.2\conf\openssl.cnf" -out <csr file name>.csr

This command will prompt for the following X.509 attributes of the certificate:

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:California
Locality Name (eg, city) []:Mountain View
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Symantec Corporation
Organizational Unit Name (eg, section) []:SSL Department
Common Name (eg, YOUR name) []:apache.netsure.net
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Country Name: Use the two-letter code without punctuation for country, for example: US or CA.

State or Province: Spell out the state completely; do not abbreviate the state or province name. For example: California, not CA.

Locality or City: The Locality field is the city or town where the organization is headquartered spelled in full. For example: Mountain View, not Mt. View

Company: If the company or department has an &, @, or any other symbol using the shift key in its name, the symbol must be spelled out or omitted, in order to enroll. Example: XY & Z Corporation would be XYZ Corporation or XY and Z Corporation.

Organizational Unit: This field is optional; but can be used to help identify certificates registered to an organization. The Organizational Unit (OU) field is the name of the department or organization unit making the request. To skip the OU field, press Enter on the keyboard.

Common Name: The Common Name is the Host + Domain Name. It looks like "www.company.com" or "company.com". SSL Provider certificates issued to the www subdomain will also include the base domain as a Subject Alternative Name.

NOTE: Please do not enter an email address, challenge password or an optional company name when generating the CSR.

Once the CSR has been created proceed to enroll for the certificate. When prompted for the CSR, please open the file with a plain-text editor (Notepad, Vi) and copy/paste the full text of the CSR into the order form.

Step 3

Backup the private key

SSL Provider recommends backing up the .key file and storing of the corresponding pass phrase. A good choice is to create a copy of this file onto a diskette or other removable media. While backing up the private key is not required, having one will be helpful in the instance of server failure.

Apache Installation Instructions

Step 1

Obtain and install the SSL certificate

a. Download the server & intermediate certificate.
b. Copy the certificate files into the directory that will hold the

certificates. For example: /usr/local/ssl/crt/

Step 2

Configure the server

Note: Some instances of Apache contain both a **httpd.conf** and **ssl.conf file**. Please update the httpd.conf or the ssl.conf with the below directives. Do not enter both as there will be a conflict and Apache may not start.

Note: Apache version 2.4.8 allowed the server certificate to be concatenated with the intermediate. Please click here for documentation on this alternative installation process.

- Open the .conf file in a plain text editor such as VI or Notepad.
- In the Virtual Host section of the **httpd.conf or ssl.conf** file, locate or add the directives below.

  a. SSLCertificateFile /[path]/[server certificate file]
  b. SSLCertificateKeyFile /[path]/[private key file]
  c. SSLCertificateChainFile /[path]/[intermediate certificate file]

Note: Some versions of Apache will not accept the **SSLCertificateChainFile** directive. Try using **SSLCACertificateFile** instead.

```
#   Server Certificate:
#   Point SSLCertificateFile at a PEM encoded certificate.  If
#   the certificate is encrypted, then you will be prompted for a
#   pass phrase.  Note that a kill -HUP will prompt again.  Keep
#   in mind that if you have both an RSA and a DSA certificate you
#   can configure both in parallel (to also allow the use of DSA
#   ciphers, etc.)
SSLCertificateFile /usr/local/ssl/crt/public.crt

#   Server Private Key:
#   If the key is not combined with the certificate, use this
#   directive to point at the key file.  Keep in mind that if
#   you've both a RSA and a DSA private key you can configure
#   both in parallel (to also allow the use of DSA ciphers, etc.)
SSLCertificateKeyFile /usr/local/ssl/private/private.key

#   Server Certificate Chain:
#   Point SSLCertificateChainFile at a file containing the
#   concatenation of PEM encoded CA certificates which form the
#   certificate chain for the server certificate. Alternatively
#   the referenced file can be the same as SSLCertificateFile
#   when the CA certificates are directly appended to the server
#   certificate for convinience.
SSLCertificateChainFile /usr/local/ssl/crt/intermediate.crt
```

1. The VirtualHosts in the httpd.conf file should be configured as follows:
   <VirtualHost [IP ADDRESS]:443>
   ServerAdmin [admin email address]
   DocumentRoot /[path]/[to document root]
   ServerName [site address]
   ErrorLog /path/[error log]
   SSLEngine on
   SSLProtocol all
   SSLCertificateFile /[path]/[server certificate file]
   SSLCertificateKeyFile /[path]/[private key file]
   SSLCertificateChainFile /[path]/[intermediate certificate file]
   ServerPath /[path]
   <Directory "/path]">
   </Directory>
   </VirtualHost>
2. Save the httpd.conf or ssl.conf file and restart Apache. This can most likely do so by using the apachectl script:
   apachectl stop
   apachectl startssl

Nginx Server: Certificate Signing Request (CSR) Generation Instructions

To generate a CSR on Nginx, please do the following:

1. **Login to your server via your terminal client (ssh). The first step will be generating the private key. At the prompt, type:**
   openssl genrsa -out [private-key-file.key] 2048
2. Once the private key has been generated, run the command below to generate the CSR.
   Openssl req -new -key [private-key-file.key] -out [CSR-file.txt]
   - **Country Name (C)**: Use the two-letter code without punctuation for country, for example: US or CA.
   - **State or Province (S)**: Spell out the state completely; do not abbreviate the state or province name, for example: Oregon.
   - **Locality or City (L)**: The Locality field is the city or town name, for example: Eugene.
   - **Organization (O)**: If your company or department has an &, @, or any other symbol using the shift key in its name, you must spell out the symbol or omit it to enroll, for example: XY & Z Corporation would be XYZ Corporation or XY and Z Corporation.
3. - **Organizational Unit (OU)**: This field is the name of the department or organization unit making the request.
   - **Common Name (CN)**: The Common Name is the Host + Domain Name. For example www.bbtest.net or secure.bbtest.net
   - **Optional Fields**: When prompted, please do not enter your email address, challenge password or an optional company name when generating the CSR. Pressing Enter/Return will leave these fields blank.
4. Your CSR file will then be created.
5. Proceed to Enrollment and paste the CSR in the enrollment form when required.

Nginx Server Installation instruction

Step 1

Obtain the SSL Provider Certificate

Download SSL Provider Certificate. Please select "**Other**" / "**X.509**" as the platform when downloading.

Step 2

Concatenate the SSL and Intermediate CA Certificate

a. You need to combine the **Server certificate (SSL_Certificate.crt)** file and the **Intermediate CA Certificate (intermediateCA.crt)** into a single concatenated file
b. To get a single concatenated file out of the Intermediate CA and the SSL Certificate run the following command: **cat intermediateCA.crt >> SSL_Certificate.crt**

Step 3

Edit the Nginx virtual hosts file

a. Open your Nginx virtual host file for the website you are securing.
   NOTE: If you need your site to be accessible through both secure (https) and non-secure (http) connections, you will need a server module for each type of connection.
b. Make a copy of the existing non-secure server module and paste it below the original.
   Then add the lines in bold below:
   server {
   listen 443;
   ssl on;
   ssl_certificate /etc/ssl/[concatenated file];
   ssl_certificate_key /etc/ssl/[private key file];
   server_name your.domain.com;
   access_log /var/log/nginx/nginx.vhost.access.log;
   error_log /var/log/nginx/nginx.vhost.error.log;

```
location / {
root /home/www/public_html/your.domain.com/public/;
index index.html;
}
}
```

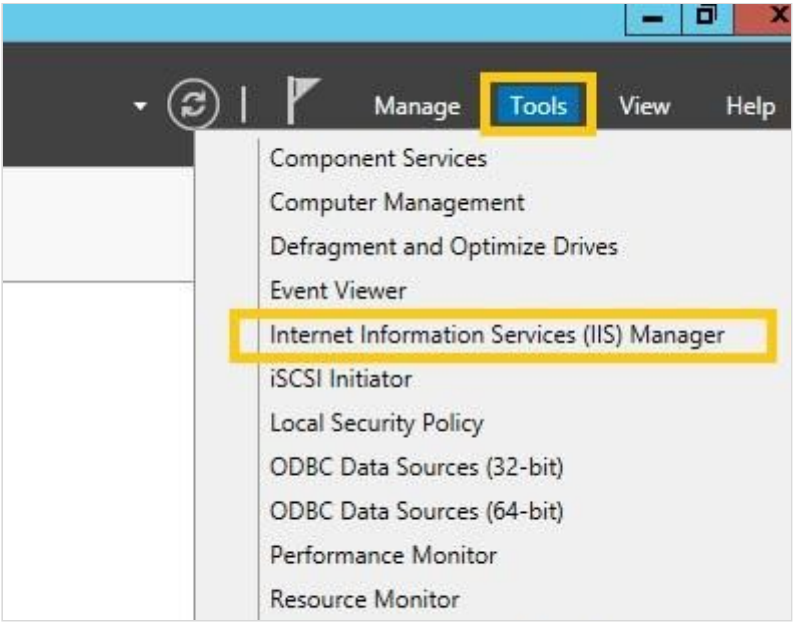
c. Adjust the file names to match your certificate files:
ssl_certificate should be your concatenated file created in **Step 3**
ssl_certificate_key should be the key file generated when you created the CSR**.**

d. Restart Nginx. Run the following command to restart Nginx:
sudo /etc/init.d/nginx restart.

Microsoft 2012 Server - IIS 8 Certificate Signing Request (CSR) Generation Instructions

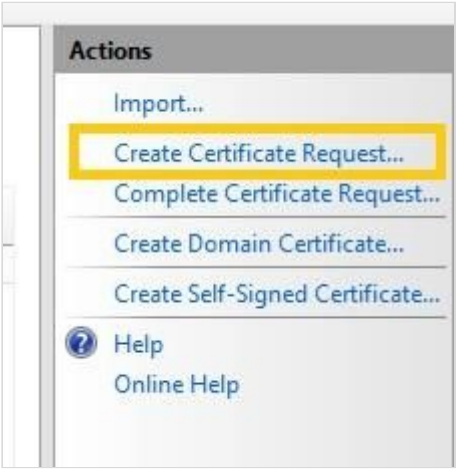
To generate the Certificate Signing Request (CSR) file, perform following steps:

1. Start the IIS by selecting **Tools** > **Internet Information Services (IIS) Manager.**



2. Select **Server Certificates**

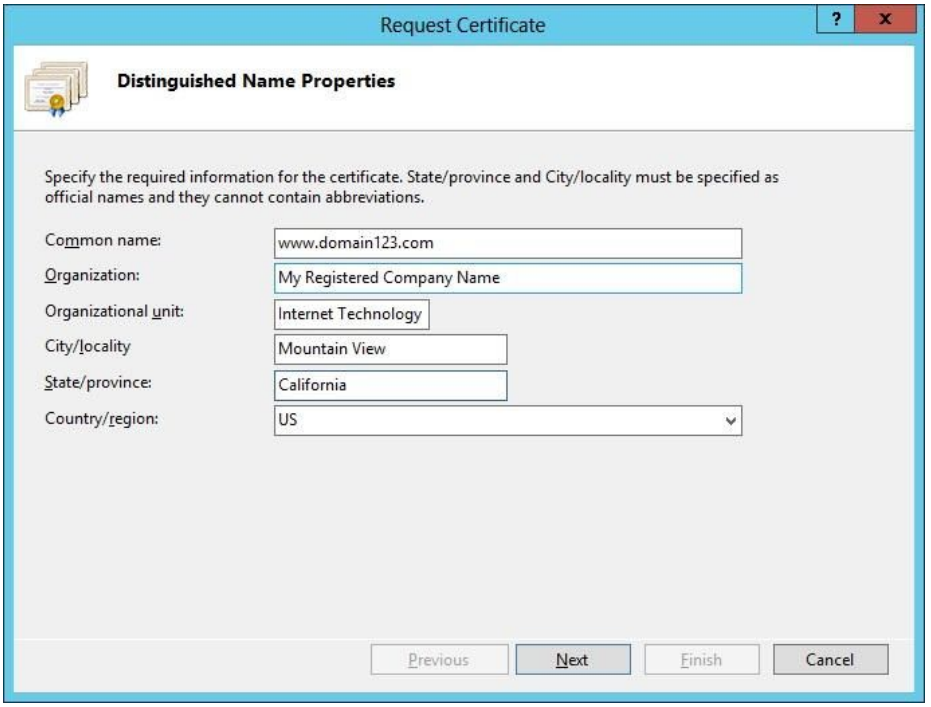
3. **Select Create Certificate Request** on the right side



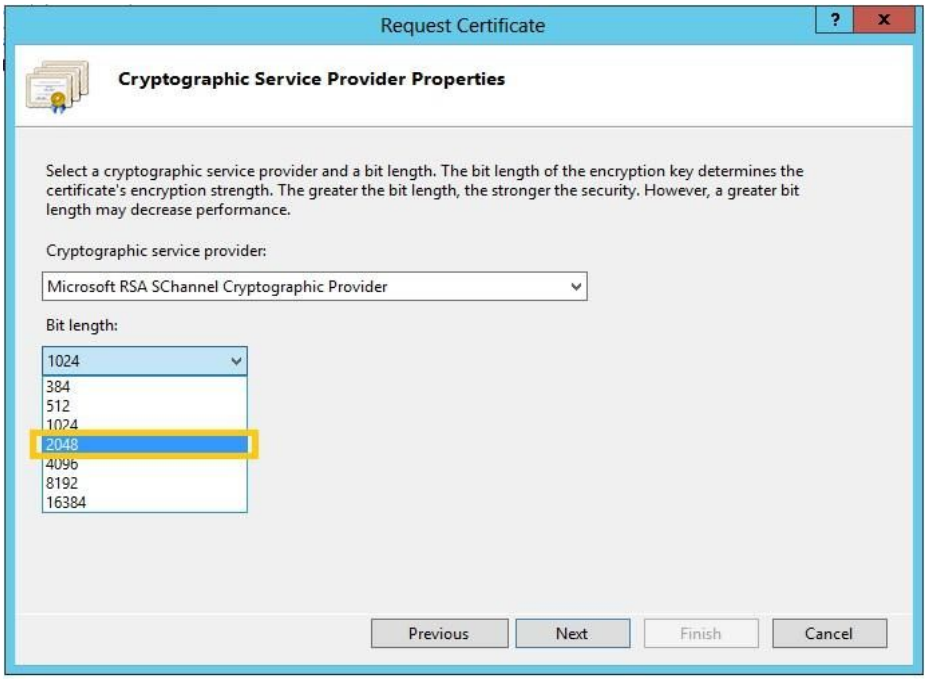
4. Complete in full all the required fields in the new page that will appear, without any abbreviation, except the country code.

   Common Name: The Fully Qualified Domain Name that the certificate will be issued to and secure.
   Organization: The Registered Organizational Name the certificate belongs to.
   Organizational Unit: The Department within the Organization
   City/locality: The Business registered location (not the actual server location)
   State/province: The Business registered state or province

Country/region: The two letter ISO country code

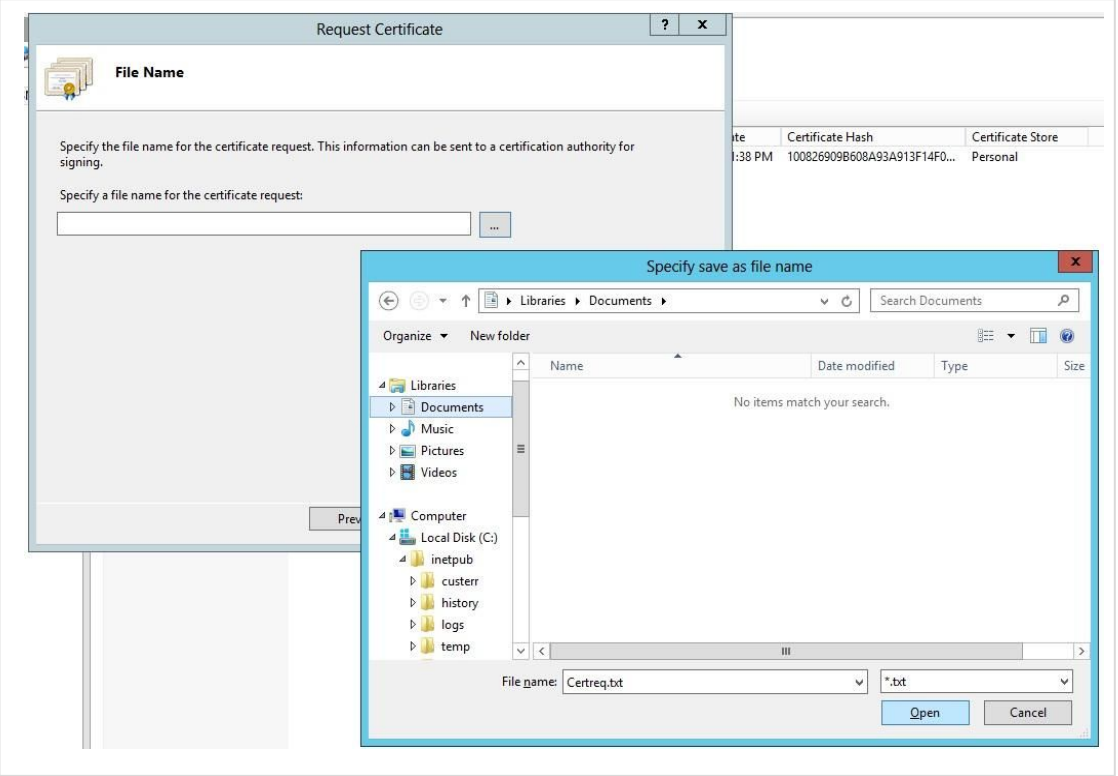


5. Select **Next**
6. **Select 2048** or **4096** from the **Bit Length:** drop down list

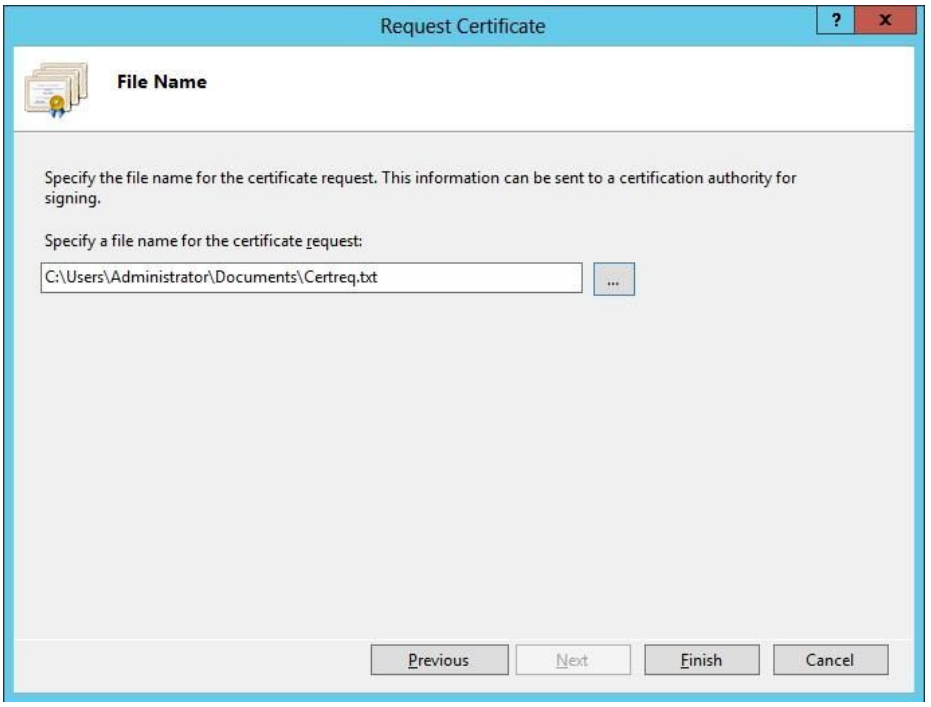


7. Select **Next**

8. Specify the file name and location for the certificate request file



9. Select Finish



10. Copy the saved CSR when requesting a certificate including the:

    -----BEGIN NEW CERTIFICATE REQUEST-----
    and
    -----END NEW CERTIFICATE REQUEST-----

11. Once the CSR has been created, log into the account and submit the CSR with an order.

Certificate Installation Instructions

Step 1

Download the certificate.

1. Download SSL Provider Certificate. Please select **Microsoft** as the server platform, or the closest matching version of IIS listed. All versions of IIS will use the same format.

Step 2

Install SSL Provider Certificate

1. **Go to Start > Administrative Tools > Internet Information Services (IIS) Manager**
2. **From the left menu, click the corresponding server name where the certificate will be installed**
3. **In the Features** pane (middle pane), under **Security**, double-click **Server Certificates**
4. **From the Actions** pane (right pane), select **Complete Certificate Request**
5. **Provide the location of the certificate file and a Friendly Name. Update the File Types** in the **Open** dialog box to **All Files (*.*)** to select the .P7B certificate file. The Friendly Name is a reference name for quick identification of the certificate for the Administrator
6. Be sure that the **Personal** store is selected, then click **OK**

NOTE: At this point the server may respond with one of two known error messages referenced below. If an error is received, click one of the links below to complete the installation. If no error is reported, proceed to **Step 3.**

CertEnroll::CX509Enrollment::p_InstallResponse:ASN1 bad tag value met. 0x8009310b (ASN: 267)

Cannot find the certificate request associated with this certificate file. A certificate request must be completed on the computer where it was created.

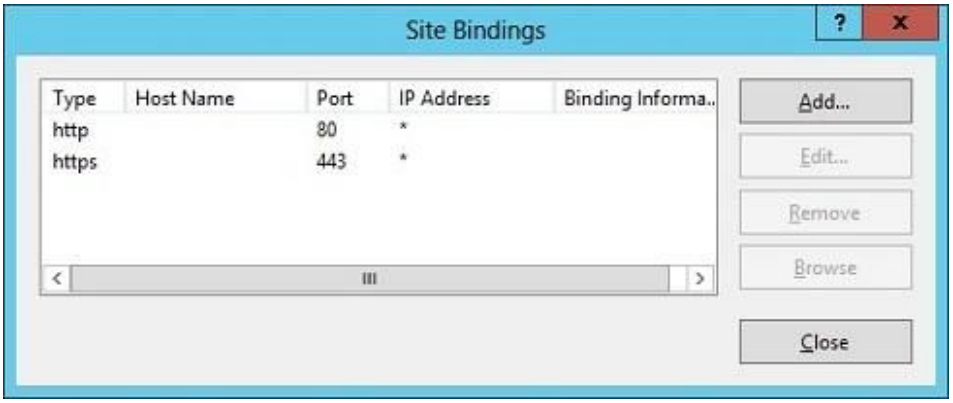
Step 3

Binding the SSL Provider certificate to the web site

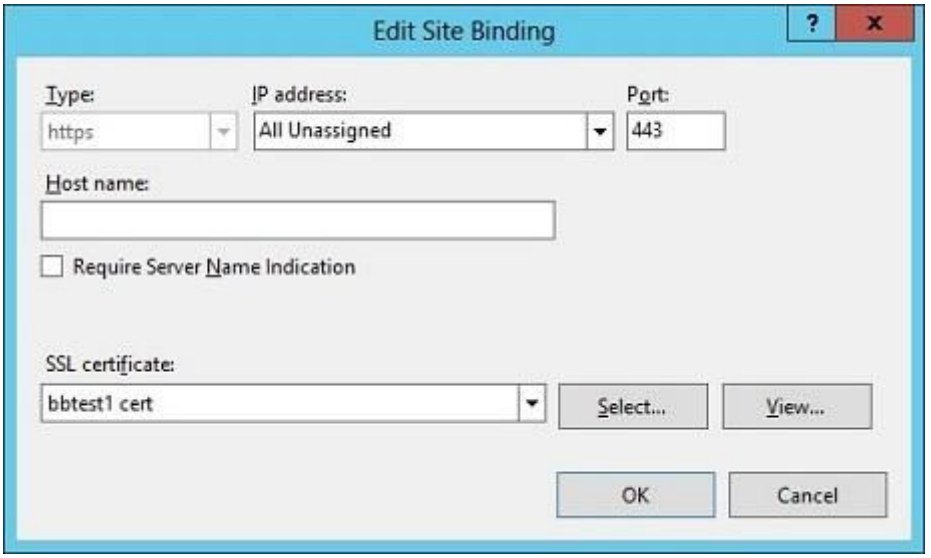
1. **From the Connections** pane on the left, expand the **Sites** folder
2. Select the appropriate web site
3. From the **Actions** pane on the right, click on **Bindings**

4. In the Site Bindings window, If there is no existing https binding, choose **Add** and change **Type** from **HTTP** to **HTTPS**

NOTE: If there is an https binding, select it and click **Edit**



5. From the **SSL Certificate** drop down, select the friendly name for the SSL certificate that was used during installation



6. Click **OK**

Step 4

Verify certificate installation

1. To verify the SSL Provider certificate installation, use the RapidSSL Installation Checker.
2. In some rare cases, a restart of IIS or a reboot of the server may be necessary in order for the changes to take effect.

Microsoft IIS 8 Installation instructions

Step 1

Download the SSL Provider certificate.

1. Download SSL Provider Certificate. Please select **Microsoft** as the server platform, or the closest matching version of IIS listed. All versions of IIS will use the same format.

Step 2

Install SSL Provider Certificate

1. Go to Start > Administrative Tools > Internet Information Services (IIS) Manager
2. From the left menu, click the corresponding server name where the certificate will be installed
3. In the Features pane (middle pane), under **Security**, double-click **Server Certificates**
4. **From the Actions** pane (right pane), select **Complete Certificate Request**
5. **Provide the location of the certificate file and a Friendly Name. Update the File Types** in the **Open** dialog box to **All Files (*.*)** to select the .P7B certificate file. The Friendly Name is a reference name for quick identification of the certificate for the Administrator
6. Be sure that the **Personal** store is selected, then click **OK**

NOTE: At this point the server may respond with one of two known error messages referenced below. If an error is received, click one of the links below to complete the installation.

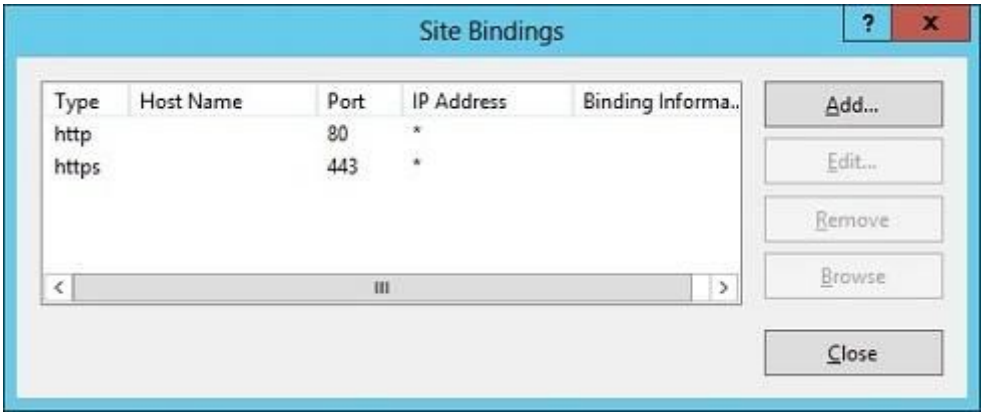
Step 3

Binding the SSL Provider certificate to the web site

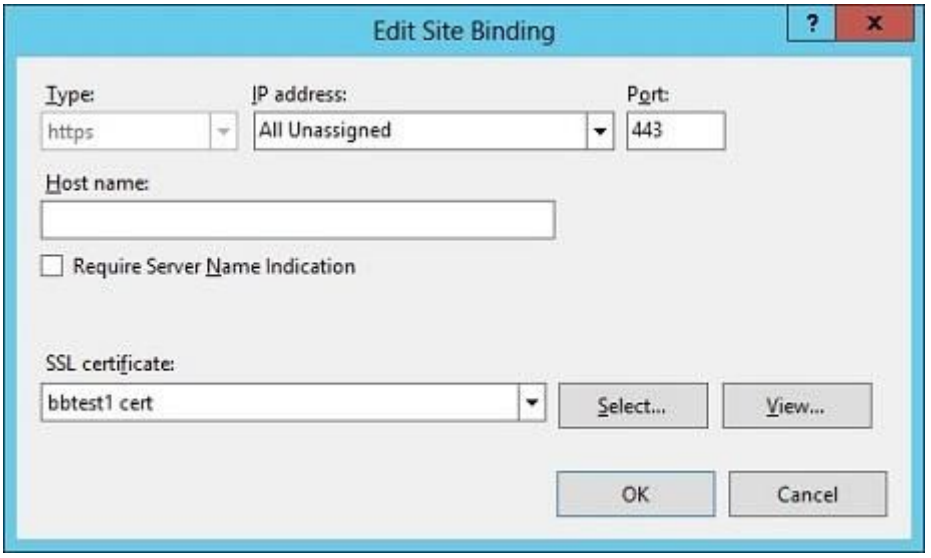
a) From the **Connections** pane on the left, expand the **Sites** folder
b) Select the appropriate web site
c) From the **Actions** pane on the right, click on **Bindings**

1. In the Site Bindings window, If there is no existing https binding, choose **Add** and change **Type** from **HTTP** to **HTTPS**

   NOTE: If there is an https binding, select it and click **Edit**



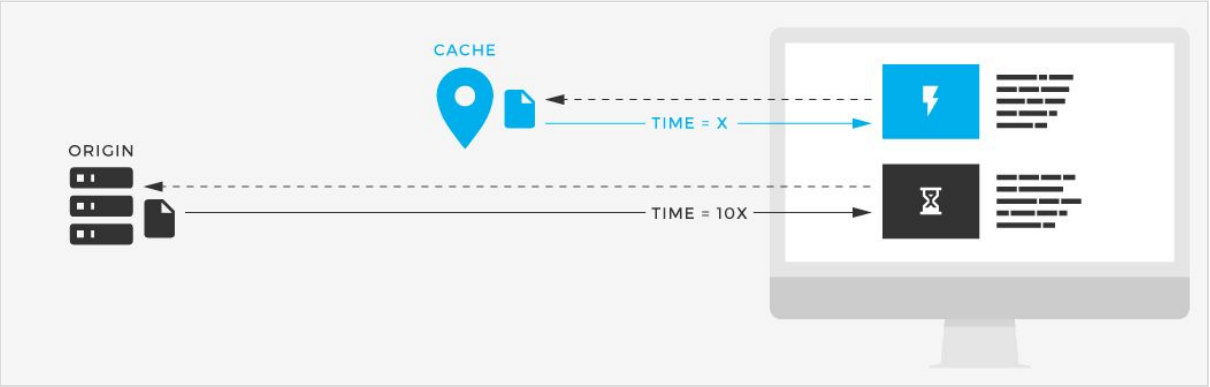
2. From the **SSL Certificate** drop down, select the friendly name for the SSL certificate that was used during installation



3. Click **OK**

Enable Server Caching

Page load time is a key performance indicator for any web service and directly impacts conversions and UX. If a website or application is only available from a single origin server, users can face delays of hundreds of milliseconds or more when loading content.

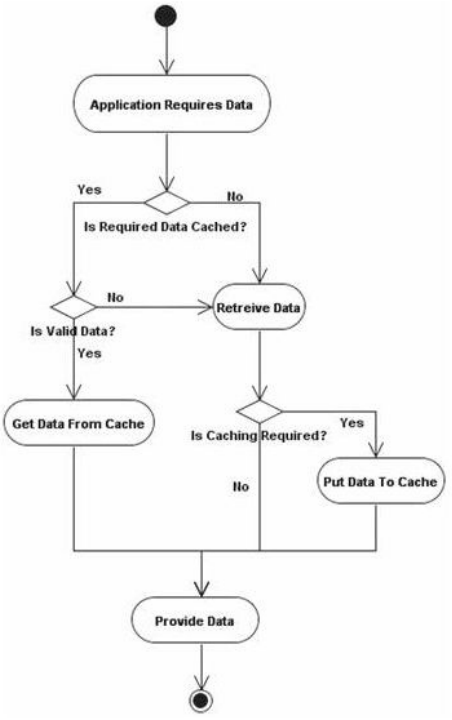


A web cache (or HTTP cache) is an information technology for the temporary storage (caching) of web documents, such as HTML pages and images, to reduce bandwidth usage, server load, and perceived lag. The cached version of resources are refreshed every time the content of a web page is refreshed.

How Caching Works

- A user clicks a link to a web page containing static content (images, videos, stylesheets, etc).
- The Application server checks if the information has been cached, if the content has been cached, it is directly forwarded to the user. If the information has not been cached, It is processed and a copy is sent to the user while a copy is stored on the server for future requests.

Caching Activity Flow Diagram



Caching Techniques

- **Web content caching:** Content caching is a mechanism for temporarily storing a web page. Doing this will prevent things like server lag, bandwidth usage, and help users to surf the web faster.
- **Network caching:** Network caching is the technique of keeping frequently accessed information in a location close to the requester. i.e. a web cache stores Web pages and content on a storage device that is physically or logically closer to the user.

Reference Link

[Caching Patterns and Implementation]

Database optimization

Page load speed is important, as it affects the user-experience as well as Google page rank when it gets too slow. There are so many variables to account for when trying to improve page load, including page download weight (including all the various assets such as images, javascript and CSS), network latency, browser cache and server headers, server load (requests per second and memory and CPU usage) among others. We're going to suspect the database as the culprit for the purposes of this article, but you should first observe the complete picture before deciding on what to optimize.

Aside from page load speed, a busy database can affect the rest of the server as well, meaning parts that don't use the database or have very fast running queries could start to slow down.

Profile first, optimize last

The basic rule of optimization is to never assume - always verify, using actual data. The process of collecting performance metrics and determining performance issues is called profiling. We want to know whether database performance is responsible for a significant part of our page load time.

We want the time of how many queries are taking to run. There are plenty of timing solutions out in the open - such as PEAR_Benchmark, that there is simply no need to build your own unless you want the exercise. The concept is simple - store microtime() values before and after the query for later observation, and the difference would be the timing of the query with good accuracy.

If you are using a database abstraction class (and you should), incorporating a timer to profile every query should be a piece of cake - so no need to hunt down every query and modify the code around it as suggested in the SM article. Wrap the query method of your abstract class with the timer and use the queries as the keys in the timing array. We used the Zend Framework as example and for our current demo,

and it comes with a built-in support for profiling which makes it a breeze to get started.

Web hosting: Using right hosting plan? (Refer "Hosting Plan" section for more details.)

Requirements

SQL access with elevated privileges

References

- Top 10 steps to optimize data access in SQL Server
- Database optimization techniques you can actually use

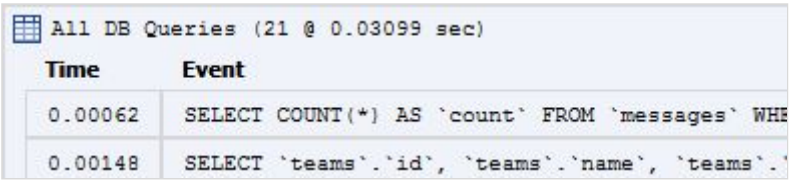
Example code using Zend_Db_Profiler

```
$db = Zend_Db::factory('PDO_MYSQL', $config); //Set up the database object
$db -> getProfiler()->setEnabled(true); // turn on profiler

//Queries are performed on the page
//...

// Where we want to show the results
$profiles = $profiler -> getQueryProfiles(); //An array of all the query profiling
```

If you are using Firebug (which you should if you are running Firefox) you can install the FirePHP plugin for completely unobtrusive output. The Zend Framework comes with a FirePHP profiler that can send query results directly into your Firebug console.

```
$profiler = new Zend_Db_Profiler_Firebug('All DB Queries');
$profiler -> setEnabled(true);
$db -> setProfiler($profiler);

//Queries are performed on the page
//...
// No need to output, query profiles will appear in your Firebug console
```

Pretty convenient. We can go over page by page without disturbing content or messing up sessions and even run this in a production environment, provided we load the profiler only for specific users. The output looks something like this:


All DB Queries (21 @ 0.03099 sec)

Time	Event
0.00062	SELECT COUNT(*) AS `count` FROM `messages` WHE
0.00148	SELECT `teams`.`id`, `teams`.`name`, `teams`.`

It's very important to profile using a relevant dataset. If you profile on a development machine that has very different data (and probably much smaller tables) than your production machine, you will get very different results. You should create a test machine that resembles your live dataset as much as possible to get relevant data (as I've shown in an old article about profiling).

Another important note is to avoid looking at cached results. MySQL will cache certain queries - so verify the results of your profiling by running the queries while avoiding caches using SQL_NO_CACHE and other means. Compare the first run with subsequent runs of the same query to be sure you are seeing non-cached results.

Aside from profiling the queries in real time, we can also profile queries that are used by daemons and cron jobs and log the results to a file. MySQL has a built-in feature in MySQL that can log slow queries for us while the database daemon is running. As of MySQL 5.1.21 we can get microsecond timing on queries (previously only one-second jumps were supported) so we can get very good measurements with the slow-query log.

The slow query log should be used for monitoring and be checked periodically for possible problems. The rate at which the log fills out also gives an indication of how much your database is slowing down over time and how much time you have left before you need to optimize.

Optimizing performance

Suppose we found out some problematic queries on slow pages. There are 4 basic ways to optimize query performance:

- Rewrite the queries
- Change indexing strategy
- Change schema
- Use an external cache

Examining query execution plans

Before trying to optimize a slow query, we need to understand what makes it slow. For this purpose MySQL has a query examination tool called EXPLAIN. Add the reserved word 'EXPLAIN' at the beginning of your query to get the execution plan for the query. The execution plan literally 'explains' to us what the database is doing to optimize the query. The MySQL manual has a full reference guide to the different values that appear in the plan, and you can see a full walkthrough of using EXPLAIN to optimize a query in this slideshow on slideshare (as well as in the profiling article I linked to earlier).

This is a very useful tool, but like all other tools it should be used while being aware of its limitations.

Common optimizations

1. Looping queries

The most basic performance issues often will not be the fault of the database itself. One of the most common mistakes is to query in a loop without need. Most likely looped SELECT queries can be rewritten as a JOIN -

(We use Zend Framework syntax since we already assumed we are using a database abstraction class)

```
$query = 'SELECT id,name FROM categories';
$rows = $db -> fetchAll($query);
foreach($rows as $row) {
```

```
    $query = 'SELECT id,name FROM sub_categories WHERE category_id=' . (int) $row['id'];
  $subCategories = $db -> fetchAll($query);
  //...
}
```

Inserting and updating rows in a loop can have major overhead as well, and those queries are generally slower than simple SELECT queries (since indexes often need to be updated) and they affect the performance of other queries since they use table / row locks while the data is written (this differs depending on the table engine). I wrote an article almost two years ago on multiple row operations that covers how to rewrite looped INSERT / UPDATE queries and includes some benchmarks to show how it improves performance.

2. Picking only needed columns

It is common to see a wildcard used to pick all columns ('SELECT * FROM ... ') - this however, is not efficient. Depending on the number of participating columns and their type (especially large types such as the TEXT variants), we could be selecting much more data from the database than we actually need. The query will take longer to return since it needs to transfer more data (from the hard-disk if it doesn't hit the cache) and it will take up more memory doing so.

Picking only the needed columns is a good general practice to use, and avoids those problems.

3. Filtering rows correctly and using indexes

Our main goal is to select the smallest amount of rows we need and doing so in the fastest way possible. We want to filter rows using indexes, and in general we want to avoid full table scans unless it is absolutely needed (aside from edge cases where it actually improves performance). The MySQL manual has some great information on optimizing the WHERE clause, and I'll dive into a bit more detail -
Filtering conditions include the WHERE, ON (for joins) and HAVING clauses. As much as possible, we want those clauses to hit indexes - unless we are selecting a very large amount of rows, index lookup is much faster than a full table scan. Those clauses should be used along with the LIMIT clause if relevant to filter the amount of rows / data returned by the query. The LIMIT clause itself can lend some important optimizations for queries if used correctly.
Since our goal is to hit indexes with our WHERE clause, an important rule is to avoid using calculations there. When the filtering condition has to be calculated for each row, the WHERE clause cannot use an index.

Example - fetching users created in the last 4 weeks:

1. **SELECT id,name FROM users WHERE created - NOW() < INTERVAL 4 WEEK**

Since the value of the `created` column changes from row to row, we now have a calculation in the left-hand side of the condition. This could be rewritten so that the calculation doesn't change and thus will only be performed once:

1. **SELECT id,name FROM users WHERE created > NOW() - INTERVAL 4 WEEK**

This query can use an index on the `created` column and should perform much better.
Another less common calculation but a very problematic one is correlated subqueries in the WHERE clause.

Selecting the lowest priced fruit from several fruit types:

1. **SELECT type, variety, price**
2. **FROM fruits**
3. **WHERE price = (**
4. **SELECT MIN(price) FROM fruits AS f WHERE f.type = fruits.type**
5. **)**

(Example taken from the excellent Xaprb article on the topic - you should read it). This query could be rewritten as JOIN, moving the subquery from the WHERE clause -

```
1. SELECT f.type, f.variety, f.price
2. FROM (
3.   SELECT type, MIN(price) AS minprice
4.   FROM fruits
5.   GROUP BY type
6. ) AS minfruits
7. INNER JOIN fruits AS f ON f.type = minfruits.type AND f.price =
   minfruits.minprice
```

Ideally, MySQL would've optimized both of those the same, but since it is usually not the case, rewriting correlated subqueries as joins is preferred.

4. Indexing correctly

Whether optimizations are needed or not is dependent on the EXPLAIN results we mentioned previously. If the execution plan indicates an index is not being used or a non-selective index has been picked, we need to understand why and change our indexing strategy accordingly (or use index hints).

MySQL can use one index per table alias in a query, so we need to plan our indexes to maximize their effectiveness. Using more indexes than is necessary can have adverse affects - as it slows down the operation of INSERT and UPDATE queries, while taking up more memory. Some indexes can even slow down performance depending on their selectivity.

If we use ordering clauses such as ORDER and GROUP BY, our indexes should often be composite indexes (indexes covering more than one column) to allow for both the filtering and ordering to use an index.

5. Picking the right engine for your data

MySQL has a pluggable engine design, which allows you to use different engine types to store your data, each with its own advantages and drawbacks. The two main engines are MyISAM and InnoDB, and the differences between them affect much

more than just performance - InnoDB is an ACID compliant transactional engine, while MyISAM sacrifices some integrity and consistency features for simplicity and performance. Having said that, MyISAM is not necessarily faster, only in some cases.

I use InnoDB for all of my table unless I need a full-text index (a MyISAM only feature), since my tables usually have a lot of write activity. InnoDB uses row-level locks which really helps performance for such use, while MyISAM uses table locks for write operations. It also optimizes write operations by treating indexes differently than MyISAM - InnoDB uses clustered indexes, so picking the right primary key is critical.

Caching

In the case that our optimizations does not yield sufficient performance benefits (either due to technical reasons or our own skill level), caching is a viable strategy to reduce database load. You should always try and optimize the database itself first, since caching will add another layer of complexity to our application.

MySQL has an internal query cache that caches results from frequently running queries if it meets certain requirements. If our queries are cached by MySQL (this can be verified by running the queries several times), there is no need to cache it - we just need to be aware that a MySQL service restart could cause a noticeable slow down while the cache is being primed again. You can read more about the dangers of the internal cache on the excellent MySQL performance blog (a must read for any serious MySQL user).

There are many caching strategies and that is the topic for another post. Common options include caching to disk (files) or caching to memory (using solutions such as a memcache or APC). Another form of caching is to the database - by de-normalizing the schema to store data that is the result of expensive to run queries.

Server tuning and beyond

Everything covered here is just the tip of the iceberg - it gets rapidly more advanced as you get dig deeper, including tuning MySQL server variables (and you should - at least the basics), the server itself (hardware / software) and using related tools such as sphinx and lucene to offload some of the work. I tried to give a good starting point and as many references as possible for getting a good start to getting your database in shape.

I linked to several excellent resources in this article, such as the MySQL manual, MySQL performance blog and Xaprb (the last two are of Percona fame - world-class experts on MySQL). I suggest you start visiting those regularly as they offer excellent advice.

Evaluating, Deleting, Updating Plugins

Evaluating Plugins

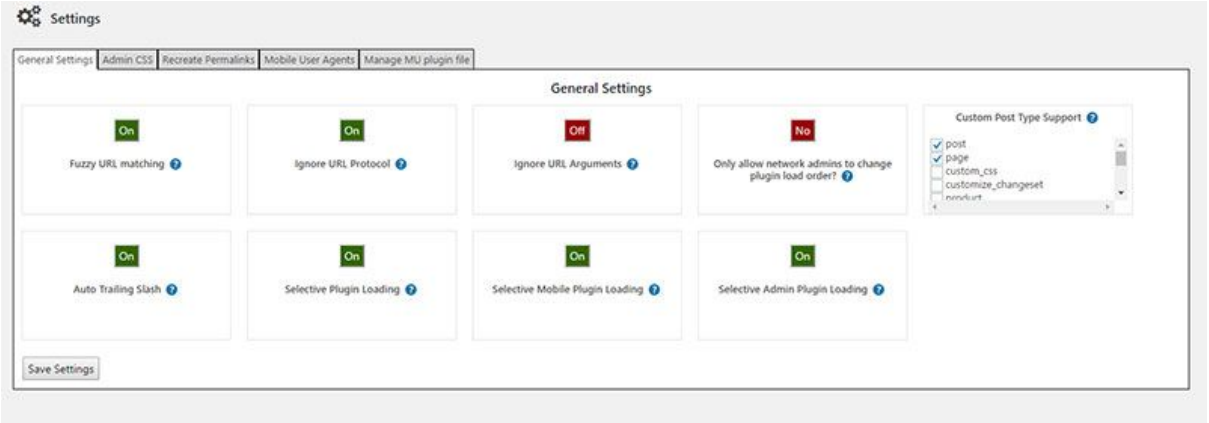
We need to ensure that the plugins are well-documented, and evaluate them against a list of pre-established criteria if required, before installing any plugin. Be wary of plugins that do far more than you need; they can end up adding substantial overhead to your page.

Requirements

CPanel / FTP / SSH / Application access with elevated privileges to modify / upload files.

Using Plugin Organizer :

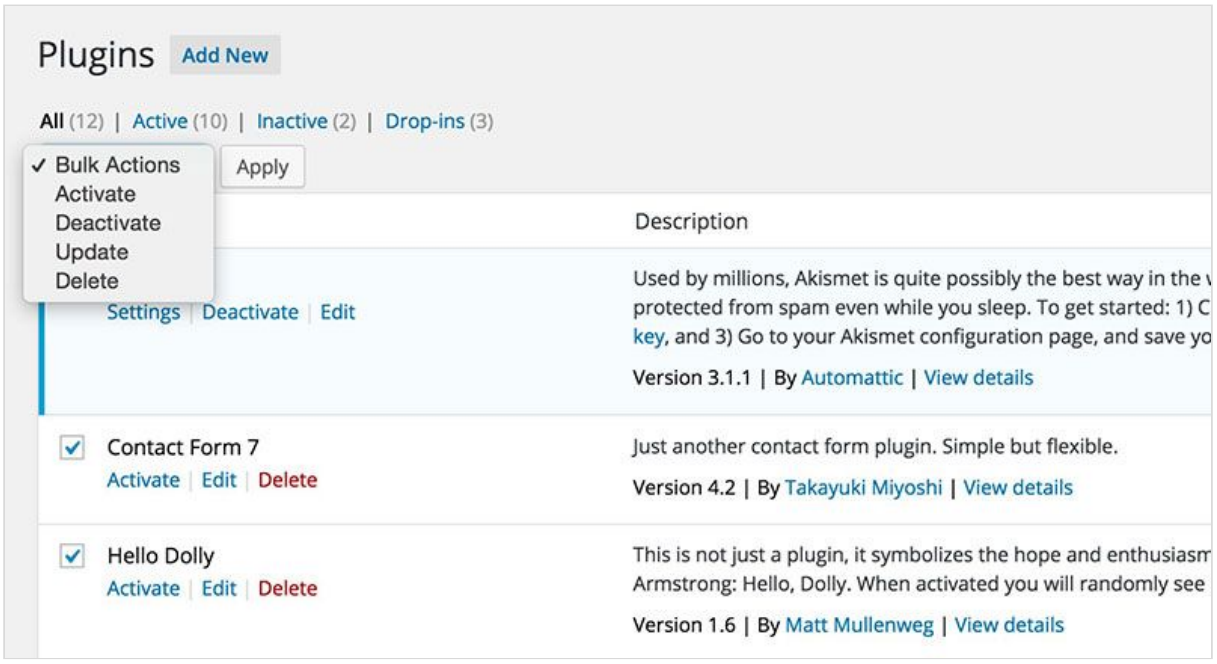
Plugin organizer is a plugin that allows you to control order that your plugins are loaded and Selectively disable plugins by any post type or wordpress managed URL.



References

- 10 Quick Ways to Clean Up and Optimize Your WordPress Site
- 7 Warning Signs Your WordPress Site Is Cluttered
- https://wordpress.org/plugins/plugin-organizer/

Deleting Plugins & Themes That You Aren't Using



Find all the plugins and themes you no longer need, then deactivate, uninstall and delete them.

Updating Themes & Updating/Replacing Plugins

After you've deactivated and deleted the unwanted plugins and themes, update all themes and plugins that you are using or are likely to use in the near future.

Delete Old Post Revisions

By deleting old revisions of posts, you can cut the size of your database down quite a bit.

While this is said not to have any real effect on the speed of your site (revisions are only accessed in the backend), making the size of your database smaller should at least help in working with it (backing it up, moving it around, etc.).

Points to consider before choosing a plugin/CMS/Application?

1. Is it an extensive plugin ecosystem?

2. The quality of plugin, usage, installation instructions and screenshots

3. Version it supports

4. Dependency of plugins

5. The ease with which the application can be configured or maintained.

6. Regular updates

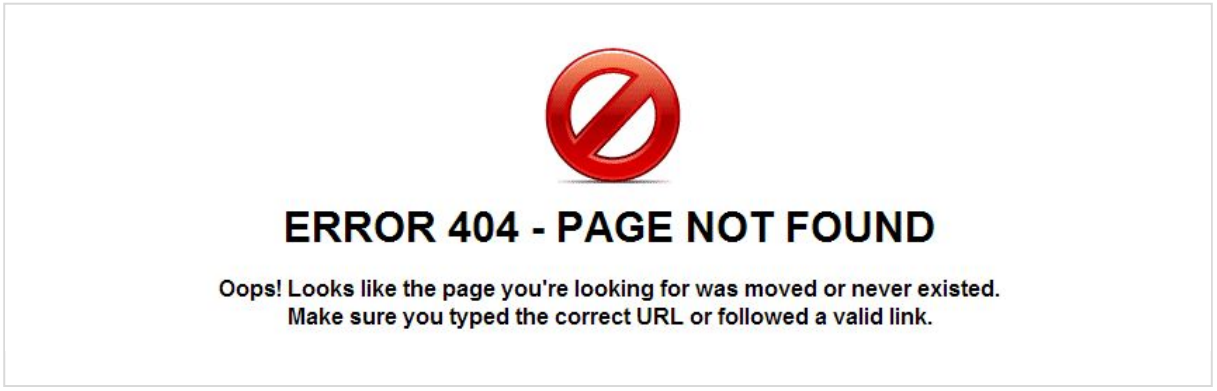
7. Author's contribution.

8. Security of plugin.

Once you choose a plugin, you'll need to add it to your page. Download the plugin, unzip it if necessary, place it within your application's directory structure, then include the plugin in your page using a script tag.

Fix broken links

There are various ways which can cause a request to be broken. The most common broken requests you'll find on most sites are 404 errors. 301's and 302's can sometimes also indicate a broken request. When a browser has to deal with a broken request it will "hang" for a short moment before it finally decides the file isn't available. In this short "hang" moment the browser will double-check if the file which causes an error is really there or not. This process creates unnecessary load time and will thus make your website slower as it could be, especially when you have many broken requests.



A broken link or dead link is a link on a web page that no longer works because the website is encountering one or more of the reasons below.

- An improper URL entered for the link by the website owner
- The destination website removed the linked web page (causing what is known as a 404 error)
- The destination website permanently moved or no longer exists
- The user has software or is behind a firewall that blocks access to the destination website
- The website owner linked to a site that is behind a firewall that does not allow outside access (such as an Intranet site or a restricted access area on a website)

Broken links can be problematic for website visitors, making them unable to access the desired resource or information. These users may decide to make use of another

site to find the necessary information elsewhere. A site that hasn't been updated or checked for a long time may suffer from link rot, which is a term used to describe a site with dozens of broken links.

Requirements

CPanel / FTP / SSH / Admin access with elevated privileges to modify / upload files.

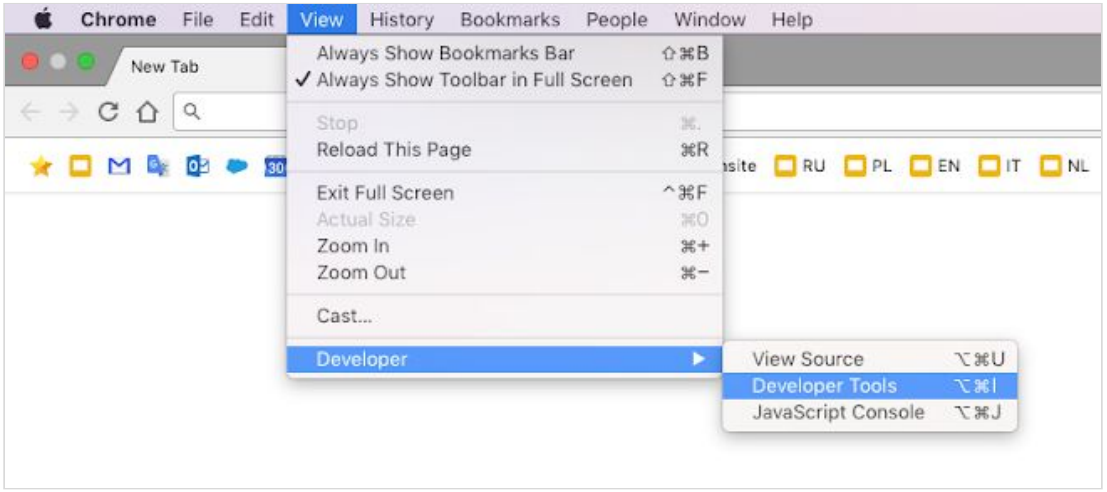
References

- Free broken link checker
- Broken Requests Test

Detailed procedure to identify and fix broken links

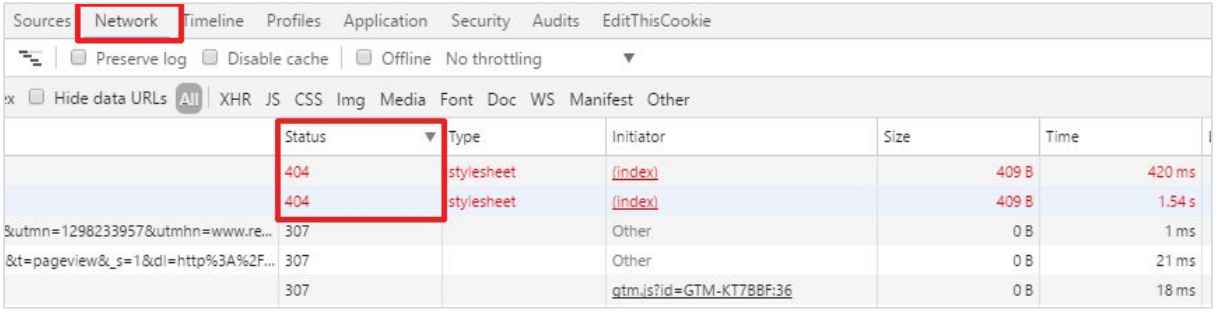
Step 1

Navigate to Chrome DevTools, Set Up Chrome DevTools, and Record a Page Load of the URL (Or any 3rd party tool Ex - http://www.deadlinkchecker.com/)



Step 2

Select “Network” settings tab and check the status of resources which are throwing 404 status.



Step 3

After using this tool and you've detected the broken requests on your page(s) you should simply replace or remove these requests from your page(s).

Accessibility

The Website should be designed to work for all people, irrespective of their physical or mental ability. When the website meets this goal, it is accessible to people with a diverse range of hearing, movement, sight, and cognitive ability. The goal here is to create websites that remove barriers that exclude people from using the web.

Requirements

- CPanel / FTP / SSH access with elevated privileges to modify / upload files.
- Access to screen reader software's / Application.

References

- Web Content Accessibility Guidelines
- Basic web accessibility

Detailed Procedure

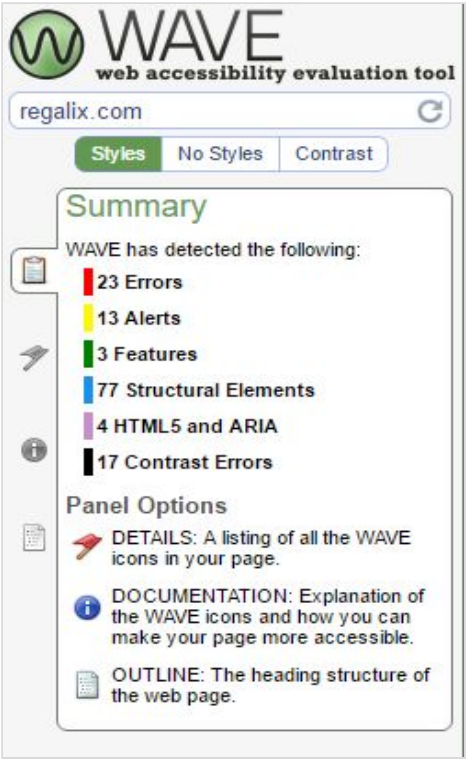
Step 1

Test if the website is accessible by visually impaired people using screen reader tools such as Wave, Jaws, achecker.ca etc.

You may refer to https://en.wikipedia.org/wiki/List_of_screen_readers for more details on screen readers.

Step 2

Analyze the report that is generated for your site and make changes accordingly. Here is how a sample report could look like:



Step 3

Review the website using the basic web accessibility suggestions and make the necessary changes.

Fixing W3 errors

The core reason to run your HTML documents through a validator is to catch unintended mistakes—mistakes you might have otherwise missed—so that you can fix them. W3 validator checks the markup validity of Web documents in HTML, XHTML, SMIL, MathML, etc.

Requirements

CPanel / FTP / SSH access with elevated privileges to modify and upload files

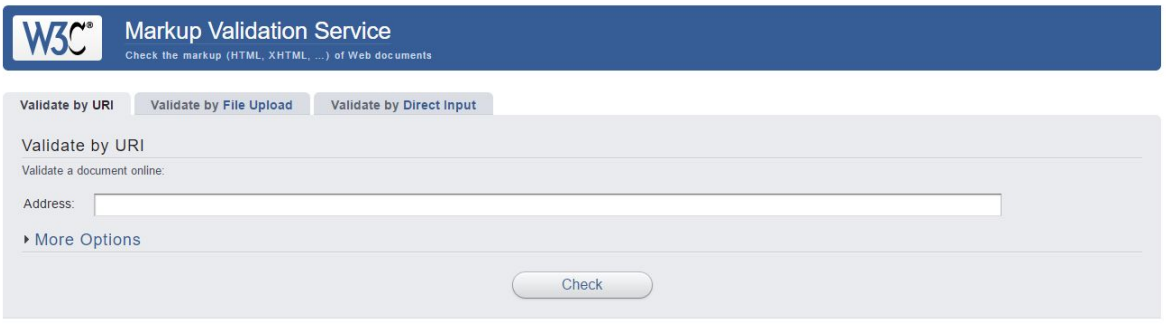
References

- Web Content Accessibility Guidelines
- Nu Html Checker

Detailed Procedure

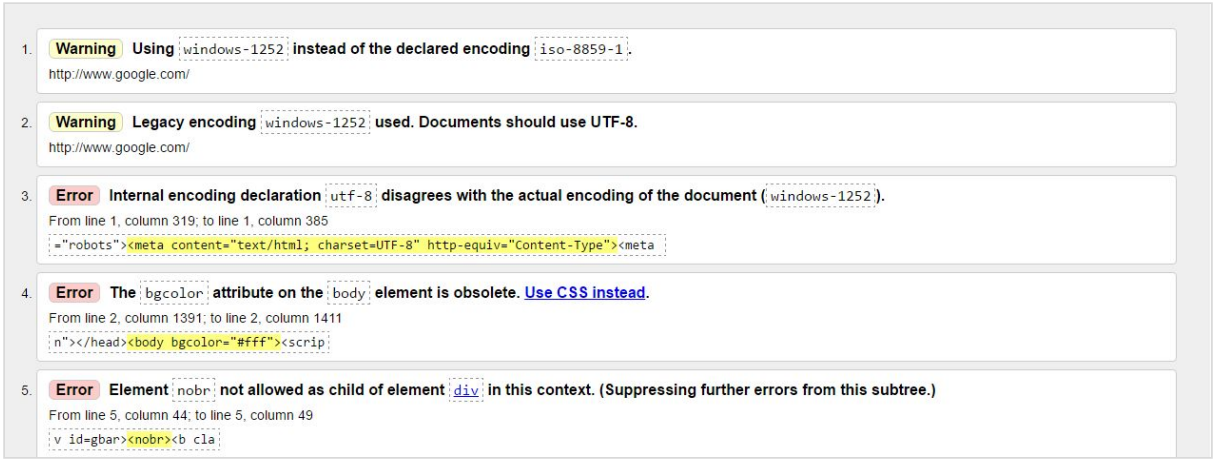
Step 1

Run website through validator.w3.org or validator.w3.org/nu/



Step 2

Analyze the error report generated by the tool



Step 3

Fix errors as per the recommendation provided by validator tool in the web page

Step 4

Perform a round of Smoke Testing to make sure there are zero negative impacts on website

Step 5

Check the website post fixing validation errors with validator.w3.org or validator.w3.org/nu/