# Page Load Optimisation Audit Report

#### Overview

This report reviews how well your site is optimised for page loading. It is becoming more important each year to ensure your content can be accessed as quickly as possible. Users become accustomed to sites that load quickly, such as Facebook and Amazon, and mobile apps, which are instantly rendered and fully functional. Then they expect the same performance on websites like yours.

When your site doesn’t meet the expectations of your users, they could leave your site and visit the site of your competition. This is particularly important to remember as more businesses go online each year.

Web optimisation is all about ensuring the code and files that make up your pages are as small as possible. This enables you to reduce the time it takes your users to download them, as well as decreasing their bandwidth usage. In addition to this, web optimisation will reduce the number of files required by your pages and prioritise the order in which they are used by browsers.

According to research by Aberdeen Group, every second that a page takes to load will decrease page views by 11%, conversions by 7%, and customer satisfaction by 16%. When your website loads faster, it can generate more business for your company.

## Table of Content

1. [Response Time](#X4fc394a10acb6ecd227b0f1d68e3ef4edb742d2)  
   1.1. [Accelerated Mobile Pages (AMP)](#X5b20f5e829fd94f89932348e7a68898e1caff8f)  
   1.2. [Server-Side Caching](#X7cfee974b09a544eae072a80d7a2a401b5051c1)  
   1.3. [Server Location](#X141a451f9a8801d11059b81ec026a588c17dada)  
   1.4. [HTTP Strict Transport Security (HSTS)](#X0e53e23fa28092bb7c149b981f96e9562c2bdb8)  
   1.5. [Content Delivery Network (CDN)](#X82eb103f722a7798b85685aa6b8e588002e4c02)  
   1.6. [Keep Alive](#X54e9da88c7cd212e32c890c039c20a7fe3aaeb0)
2. [Time to First Paint](#Xac5c86752b82cfa78863ab9bb841375614b0fab)  
   2.1. [Size of HTML Pages](#X909cebe6d9c256f6fe6a97e9334b5664f89312a)  
   2.2. [Prioritising Critical CSS](#Xab3ab6abc046b3a6c8bb74dbd8cf52a2ecda899)  
   2.3. [Server Push](#Xcd84c57821d6ffcbf7cdb8923363096cfffed2b)  
   2.4. [Unused Code](#X8e2cee313fda6146534a09049043d0a936b567a)  
   2.5. [Non-Essential Styles](#X05efb2d88d8e1d3a078d3facea4b2289a49e217)  
   2.6. [Code Minification](#X4c056573443c65186c4a4010811c3858f49bc9b)  
   2.7. [Text Compression](#Xf40f4d742c8e209f6be0c95e229828943f781a9)  
   2.8. [Async Loading of Code](#Xd37e9d9df6e5d8b4d5efeb3a7cf4a1dc228469e)  
   2.9. [Number of Requested Files](#X5718fad8082d1f7d69a248ff5378ff88179e697)  
   2.10. [HTTP/2](#X0af0e7ebfffcf58355cf387c0773a6523618b38)  
   2.11. [Code Validation](#X029f70b2ef4a1b1179414e26d04ae9df6db4790)
3. [Page Load Time](#X6e91f1af4caacc44cf058100295052bd9b1a0ef)  
   3.1. [Total Page Size](#X838ffd7e1f4562dacf9617526ff0938378af39c)  
   3.2. [Image Format](#Xd4efcf61570bd0063e2148c98cc06c00a0dab90)  
   3.3. [Image Dimension](#X04e768b016e2cef32c54bccc63fa3c1cfba7481)  
   3.4. [Image Compression](#X964a8d9372dfcfc172f466f3bb83edaf94f1d6f)  
   3.5. [Deferred Image Load](#X9854875bcf8568a9f054e69a8cd99c6161a1ae7)
4. [Time to Load the Next Page](#X37d5d3133cd75090329178fa2a0dab23edf4e87)  
   4.1. [Prefetching](#X08c15433e493fc4b9dd440883670bbab15dde6b)  
   4.2. [Client-Side Caching](#X7833f88a856d34d49ba77c47178350406dff38e)  
   4.3. [Query Strings](#X5e0121aba04fb00bce7d2993c66f518605e5b7d)  
   4.4. [Redirects](#Xa5201c36422c45160d8291b7a80dd7d1b11159a)  
   4.5. [DNS Prefetching](#X2c68ddef51435474caebecd3207ec399877a6bc)

## 1. Response Time

#### Overview

A Response Time indicates the time taken from requesting your page in a browser to getting the raw HTML file of your page with its content for the search engines to analyse. Your server response time is affected by the amount of traffic you receive, the resources each page uses, the software your server uses, and the hosting solution you use.

To improve your server response time, look for performance bottlenecks like slow database queries, slow routing, a lack of adequate memory or server caching and fix them. The optimal server response time is under 0.2 second on desktop and [under 1.3 seconds on mobile phones with 3G connection](https://www.thinkwithgoogle.com/articles/mobile-page-speed-new-industry-benchmarks.html?utm_source=moztop10&utm_medium=email&utm_campaign=moztop10&mkt_tok=eyJpIjoiT0RZMlpqWTROMll4TUdGayIsInQiOiJtQ1BGQ0NCcmt3eDBBUCtpejBZWmVcLzBlMVJOTEJqdHZZcXk0UllW).

### 1.1. Accelerated Mobile Pages (AMP)

#### Overview

Accelerated Mobile Pages are the new way of coding your pages for smartphones users. AMP is a Google initiative; with the aim of making your pages load almost instantly.

AMP use the standard HTML tags, with a few additional AMP specific tags. AMP JS library uses all performance practices, meaning there is no need to do anything else with them. Google caches your AMP on their content delivery network to ensure they load quickly from any location.

Furthermore, they provide your pages with additional visibility in a search, as Google heavily promotes sites with AMP

#### Issue

[Described detected issues supported by screenshots and code snippets, as why these things are happening to allow a client to recreate what we are see.]

#### Recommendations

[Explicit instructions on how to fix the issue supported by screenshots, code snippets with the fixed issue and mock-ups how it should look like.]

#### Reference documents

[Spreadsheets with URLs where all these issues are happening or direct links to tools that you might be using.]

### 1.2. Server-Side Caching

#### Overview

Server-side caching enables you to save your static pages locally by placing those previously requested in temporary storage. This way they can be quickly accessed and sent to a user.

Content Management Systems (CMS) such as WordPress and e-commerce servers like Magento keep each of your pages in multiple small modules. Each time a page is requested, these platforms begins assembling the most up to date version of your page and sends it to the user. This has many advantages, but also slows down the time in which the page arrives in the user’s browser.

After configuring the disk caching, your server will store pre-generated pages on its hard drive, ready to be sent to the user, instead of assembling them each time a person wants to visit them.

#### Issue

[Described detected issues supported by screenshots and code snippets, as why these things are happening to allow a client to recreate what we are see.]

#### Recommendations

[Explicit instructions on how to fix the issue supported by screenshots, code snippets with the fixed issue and mock-ups how it should look like.]

#### Reference documents

[Spreadsheets with URLs where all these issues are happening or direct links to tools that you might be using.]

### 1.3. Server Location

#### Overview

Despite the fact that most of the time we use the Internet over a wireless connection, information on the Internet has to travel through a cable from a server on which the website is hosted. It will then travel to your network transmitter or home router, before ending up on your mobile device.

The greater the distance between your user’s location and the physical location of your server on which you keep your website, the longer the files that are required to display on your page will travel, delaying your page load.

If you host your website in New York, United States and a person from Dublin, Ireland wants to visit your page, a signal travels from a device in Dublin, through a cable on the bottom of the Atlantic Ocean and then on to a server in New York. A raw HTML page is then sent back to Dublin via the same direction. Your user’s browser then checks which files are located in the HTML, before sending a request back to New York to retrieve them. Upon receiving them, it may realise that it needs a few more files, so it would then send the request again. This process is repeated until all files are received, and each time a new page is requested it will restart from the beginning.

As you can imagine, this can add an unnecessary lag time. This is why it is always better to host your website on the server that is located in the country where the majority of your customers live. This will improve your site’s page load time.

#### Issue

[Described detected issues supported by screenshots and code snippets, as why these things are happening to allow a client to recreate what we are see.]

#### Recommendations

[Explicit instructions on how to fix the issue supported by screenshots, code snippets with the fixed issue and mock-ups how it should look like.]

#### Reference documents

[Spreadsheets with URLs where all these issues are happening or direct links to tools that you might be using.]

### 1.4. HTTP Strict Transport Security (HSTS)

#### Overview

HTTP Strict Transport Security (aka HSTS) helps speed up the firs page load of your site when you have installed an SSL certificate for your domain, and a user requests an unsecured version of your URL. This can occur when a searcher types in the address of your page (e.g. example.ie) or clicks on a link to HTTP version (e.g. http://example.ie/page/). Then your server needs to correct the request and redirect the user to the secured version of the page (if your server is set up correctly).

You can avoid this delay by using HSTS which asks browsers to automatically load the HTTPS version of your site, each time someone requests a page. This will improve the loading time of your pages by 0.1-0.3 seconds. In addition, it will significantly improve your site’s security, as from now on, the user will only be able to connect to your server through the encrypted connection.

#### Issue

[Described detected issues supported by screenshots and code snippets, as why these things are happening to allow a client to recreate what we are see.]

#### Recommendations

[Explicit instructions on how to fix the issue supported by screenshots, code snippets with the fixed issue and mock-ups how it should look like.]

#### Reference documents

[Spreadsheets with URLs where all these issues are happening or direct links to tools that you might be using.]

### 1.5. Content Delivery Network (CDN)

#### Overview

Content Delivery Networks are servers spread around the world, which can be used by your site to send static HTML, CSS, JavaScript and image files that make up your pages when a user wants to see them. The time it takes to deliver all files to the user’s browser is much shorter as these servers are located closer to the user location than your own.

Furthermore, the files served from CDNs have different subdomains, increasing the number of requests the browser is able to process simultaneously. They can also protect your website from cyber-attacks by creating a first layer of protection.

There are many Content Delivery Networks your business can choose from. Two of the most popular are Cloud Flare (widely used as a free-service) and MaxCDN (premium service).

#### Issue

[Described detected issues supported by screenshots and code snippets, as why these things are happening to allow a client to recreate what we are see.]

#### Recommendations

[Explicit instructions on how to fix the issue supported by screenshots, code snippets with the fixed issue and mock-ups how it should look like.]

#### Reference documents

[Spreadsheets with URLs where all these issues are happening or direct links to tools that you might be using.]

### 1.6. Keep Alive

#### Overview

When a website takes a very long time to respond to a browser’s request, Keep Alive allows the server to ask a browser wait longer, keeping the connection alive. Without the Keep Alive enabled on your server, the browser could quite waiting for the request page, data or a file, before server was ready to send it to the client.

https://varvy.com/pagespeed/keep-alive.html

#### Issue

[Described detected issues supported by screenshots and code snippets, as why these things are happening to allow a client to recreate what we are see.]

#### Recommendations

[Explicit instructions on how to fix the issue supported by screenshots, code snippets with the fixed issue and mock-ups how it should look like.]

#### Reference documents

[Spreadsheets with URLs where all these issues are happening or direct links to tools that you might be using.]

## 2. Time to First Paint

#### Overview

The ‘time to first paint’ is one of the most important metrics of page load optimisation, as it indicates the time it takes to show a user the first element on your page.

When the user is waiting for the page to load for a time deemed longer than usual, there is a high chance they will lose patience and leave your site, often heading to the website of a competitor.

Therefore, it is good practice to display the header of your site or, at least, the raw elements from the top part of your page as soon as possible. This allows the styles and other page elements to continue to load in the background, whilst giving an indication to the user that the page is loading.

This way, more people will get to see your pages, which will increase user engagement, leads and, ultimately, customers. Therefore, it is ideal for your pages to appear on a desktop after less than 0.5 seconds and [less than 3 seconds on mobiles with a 3G connection](https://www.thinkwithgoogle.com/marketing-resources/data-measurement/mobile-page-speed-new-industry-benchmarks/).

### 2.1. Size of HTML Pages

#### Overview

The size of your page’s HTML code impacts on how fast your page will load. This is due to the fact that before a browser can start to download images, CSS and JavaScript files are required to display the page. A browser first needs to receive the HTML code of your page with links to those files.

The larger the HTML page is, the longer the delay will be when downloading the required assets. This then slows down the time it takes for elements of your page to appear to your user. Your HTML page should ideally be close to 15KB or less in size.

Your page may occasionally contain unnecessary HTML code, inline CSS and inline JavaScript, which you could move into other files and load once the HTML page is ready.

#### Issue

[Described detected issues supported by screenshots and code snippets, as why these things are happening to allow a client to recreate what we are see.]

#### Recommendations

[Explicit instructions on how to fix the issue supported by screenshots, code snippets with the fixed issue and mock-ups how it should look like.]

#### Reference documents

[Spreadsheets with URLs where all these issues are happening or direct links to tools that you might be using.]

### 2.2. Prioritising Critical CSS

#### Overview

The CSS code is responsible for laying out your pages, adding colours, font types and sizes, etc. Styles that are responsible for styling only the immediately visible part of a page after it is loaded on a screen are called the critical CSS.

You can show a page to users even 1 second faster on mobile and 0.5 seconds faster on the desktop by configuring your website to load the critical CSS first and loading remaining styles after a page loads.

#### Issue

[Described detected issues supported by screenshots and code snippets, as why these things are happening to allow a client to recreate what we are see.]

#### Recommendations

[Explicit instructions on how to fix the issue supported by screenshots, code snippets with the fixed issue and mock-ups how it should look like.]

#### Reference documents

[Spreadsheets with URLs where all these issues are happening or direct links to tools that you might be using.]

### 2.3. Server Push

#### Overview

#### Issue

[Described detected issues supported by screenshots and code snippets, as why these things are happening to allow a client to recreate what we are see.]

#### Recommendations

[Explicit instructions on how to fix the issue supported by screenshots, code snippets with the fixed issue and mock-ups how it should look like.]

#### Reference documents

[Spreadsheets with URLs where all these issues are happening or direct links to tools that you might be using.]

### 2.4. Unused Code

#### Overview

Websites are often built on frameworks and libraries, which contain thousands of lines of code to allow web developers to quickly style pages and add additional functionality. However, most sites end up using only a small percentage of that code on their pages.

Often, plugins that you have installed on your site does not know on which pages you will need to use its functionality. So they load CSS and JavaScript files on every page, increasing its size and decreasing the load time. It is recommended to configure your site in a way that only loads these files on pages on which are needed, and removes them from others.

In addition to this, when a site goes through a number of design changes over time, the site usually contains a legacy code which is no longer used. As a code base grows, and it becomes more difficult to understand what all the styles are actually for, it starts to slow down your site.

It is recommended to replace the old code with the new one rather than simply adding the new one to it, as well as reviewing your website code every 1-2 years, excluding any code from your files that is no longer used. This will ensure your website files download as fast as possible, and that browsers will not waste time processing the legacy code.

#### Issue

[Described detected issues supported by screenshots and code snippets, as why these things are happening to allow a client to recreate what we are see.]

#### Recommendations

[Explicit instructions on how to fix the issue supported by screenshots, code snippets with the fixed issue and mock-ups how it should look like.]

#### Reference documents

[Spreadsheets with URLs where all these issues are happening or direct links to tools that you might be using.]

### 2.5. Non-Essential Styles

#### Overview

Mobile, tablet, desktop and print CSS styles are often downloaded and processed by a browser each time a page is loaded. However, only some of these styles will be used on a specific type of device.

This creates an unnecessary delay when your pages load especially on mobile devices, as they use a lower network. In addition to this, their processing power is often weaker than desktops, causing them to take longer to process your website code. To resolve these issues, you should separate print and media query styles from the main style-sheet and then mark them for mobile browsers as ‘non-critical’.

Mobile phones should only download styles dedicated for such devices. Tables and desktops could download mobile styles and use them as a baseline when loading a file with styles dedicated to them, as they most likely use a faster Internet connection. Print styles should only load when a user wants to print the page.

#### Issue

[Described detected issues supported by screenshots and code snippets, as why these things are happening to allow a client to recreate what we are see.]

#### Recommendations

[Explicit instructions on how to fix the issue supported by screenshots, code snippets with the fixed issue and mock-ups how it should look like.]

#### Reference documents

[Spreadsheets with URLs where all these issues are happening or direct links to tools that you might be using.]

### 2.6. Code Minification

#### Overview

Code Minification is the process of removing unnecessary characters, comments and whitespaces from HTML, CSS and JavaScript code without changing its functionality to reduce the size of the file.

These elements are useful to people when reading the code and updating it, although to browsers they are simply unnecessary. By removing such elements from files placed on your server, you can decrease their size even by more than half, resulting in an improvement in the time it takes for them to be downloaded.

#### Issue

[Described detected issues supported by screenshots and code snippets, as why these things are happening to allow a client to recreate what we are see.]

#### Recommendations

[Explicit instructions on how to fix the issue supported by screenshots, code snippets with the fixed issue and mock-ups how it should look like.]

#### Reference documents

[Spreadsheets with URLs where all these issues are happening or direct links to tools that you might be using.]

### 2.7. Text Compression

#### Overview

You can activate a module on your server, which will start further compressing your text-based files like HTML, CSS, JavaScript, Fonts, and XML, before they are sent to your user’s browser. Once the browser receives them, it will decompress the files and display a page on the screen.

The standard GZIP compression can achieve [up to 90% reduction](https://developers.google.com/speed/docs/insights/EnableCompression) in the size of your website’s large text-based files. While, new Google’s Brotli compression offers event [20-26% higher compression](https://opensource.googleblog.com/2015/09/introducing-brotli-new-compression.html) ratios over Zopfli (GZIP). This will significantly reduce the time it takes for pages to download and display on the screen.

#### Issue

[Described detected issues supported by screenshots and code snippets, as why these things are happening to allow a client to recreate what we are see.]

#### Recommendations

[Explicit instructions on how to fix the issue supported by screenshots, code snippets with the fixed issue and mock-ups how it should look like.]

#### Reference documents

[Spreadsheets with URLs where all these issues are happening or direct links to tools that you might be using.]

### 2.8. Async Loading of Code

#### Issue

[Described detected issues supported by screenshots and code snippets, as why these things are happening to allow a client to recreate what we are see.]

#### Recommendations

[Explicit instructions on how to fix the issue supported by screenshots, code snippets with the fixed issue and mock-ups how it should look like.]

#### Reference documents

[Spreadsheets with URLs where all these issues are happening or direct links to tools that you might be using.]

### 2.9. Number of Requested Files

#### Overview

Every time a user opens one of your pages, a browser requests from your server all the files that you have attached to this page. These files can contain CSS, JavaScript, images, videos and fonts. Every time a file of this nature is needed by a page, it creates an additional HTTP request for the browser to process.

Because browsers can only request and download approximately 6 files at the same time through HTTP requests from one domain, the rest of the files will be waiting idle, therefore slowing the page load time. Ideally, your pages [should not request more than 50 files](https://www.thinkwithgoogle.com/marketing-resources/data-measurement/mobile-page-speed-new-industry-benchmarks/), and in generally the lower this number is the better.

Plugins and themes for popular content management systems (CMS), such as WordPress, tend to load by default all some of their CSS and JavaScript files on all of your pages. This creates a significant overhead, increasing the amount of code that browsers need to download and process to display your page to a user, which results in a slow page load.

Avoid sending users unnecessary files with pages that don’t need them. Additionally, whenever you install new software on your site, you should ensure its files only load on the pages on which the software is used.

#### Issue

[Described detected issues supported by screenshots and code snippets, as why these things are happening to allow a client to recreate what we are see.]

#### Recommendations

[Explicit instructions on how to fix the issue supported by screenshots, code snippets with the fixed issue and mock-ups how it should look like.]

#### Reference documents

[Spreadsheets with URLs where all these issues are happening or direct links to tools that you might be using.]

### 2.10. HTTP/2

#### Overview

HTTP/2 is the new version of HTTP1.1 protocol. Introduced in 2015, it is a set of rules that govern the data communication process between user’s browser and your server. It has dramatically improved the performance of websites and its advantages are expected to further advance over time.

The downside of HTTP/1.1 protocol is that it limits browsers to being able to request only one file at a time per connection with your server. Browsers only support six connections per domain, which means that they will only download six files simultaneously from your server to the user’s device, causing the rest to wait for their turn.

HTTP/2, on the other hand, allows browsers to get ready to start downloading all files right away over a single connection. Additionally, it offers faster encryption, header compression that improves the performance and security, as well as steam prioritisation.

It is important that your server supports HTTP/2, as it will make unnecessary to apply some additional page load optimisation technics to your site, such as:

* splitting files over multiple domains
* combining or inlining your CSS styles
* combining or inlining your JavaScript code
* combining small images into CSS sprites
* converting some images to Base64 code

#### Issue

[Described detected issues supported by screenshots and code snippets, as why these things are happening to allow a client to recreate what we are see.]

#### Recommendations

[Explicit instructions on how to fix the issue supported by screenshots, code snippets with the fixed issue and mock-ups how it should look like.]

#### Reference documents

[Spreadsheets with URLs where all these issues are happening or direct links to tools that you might be using.]

### 2.11. Code Validation

#### Overview

Modern browsers are very intelligent, meaning that even if your pages contain an invalid code, they will still usually manage to display your page correctly. However, the process of identifying and fixing the issue may slow down the displaying of your page to the user. Therefore, we recommend fixing these types of issues in order to help the browser focus on downloading and displaying your content to searchers as quickly as possible.

Your pages should adhere to W3C standards to ensure the code is easy for browsers to read and execute. Usually, only a few selected pages of each website will be tested, as many issues for the entire website can be easily fixed by simply using the page templates.

#### Issue

[Described detected issues supported by screenshots and code snippets, as why these things are happening to allow a client to recreate what we are see.]

#### Recommendations

[Explicit instructions on how to fix the issue supported by screenshots, code snippets with the fixed issue and mock-ups how it should look like.]

#### Reference documents

[Spreadsheets with URLs where all these issues are happening or direct links to tools that you might be using.]

## 3. Page Load Time

#### Overview

Page load time outlines the length of time taken from the user clicking on a link to your page or entering your URL in the browser, to when all required files are downloaded, and the page is displayed. Usually, the page will appear on the screen somewhere in the middle of that process.

Google aims to load all pages in under 1 second, while the industry standard is less than 2 seconds on desktops. On mobile devices with a 3G connection, you should aim for no longer than 7 seconds. However, sometimes those targets may be difficult to reach due to the complexity of a page.

This is the intermediate state between Time to First Paint and Page Load Completion, which is called Time to First Meaningful Content. It’s time it takes the browser to show the first portion of the content that the user was looking for. This is the content Above the Fold, part of the page visible without the content.

### 3.1. Total Page Size

#### Overview

Page size is one of the main factors that will influence how fast the page will load. The larger the resources needed to load your page, the longer it will take a user to download them. This is especially important on a mobile phones on which [1.49MB takes seven seconds to load using a fast 3G connection](https://www.thinkwithgoogle.com/marketing-resources/data-measurement/mobile-page-speed-new-industry-benchmarks/).

Ideally, your pages should not be larger than 500 KB. This will help your pages to load faster on desktops and smartphones.

#### Issue

[Described detected issues supported by screenshots and code snippets, as why these things are happening to allow a client to recreate what we are see.]

#### Recommendations

[Explicit instructions on how to fix the issue supported by screenshots, code snippets with the fixed issue and mock-ups how it should look like.]

#### Reference documents

[Spreadsheets with URLs where all these issues are happening or direct links to tools that you might be using.]

### 3.2. Image Format

#### Overview

Ensuring images are in the correct format can further reduce their size. Using the wrong format could make your images larger in file size than necessary. By applying the following rules, you can help your users avoid unnecessary delays.

On Chrome and Opera your site should serve user images in WebP format, which makes your PNGs and JPEGs an average of 25-35% smaller. This can significantly improve the speed they appear on the screen. On all other browsers, JPEGs should be displayed for images; PNGs should be used instead of GIFs for simple graphics and those with a transparent background. For animations you should try to use CSS3 styles. Whenever this is not possible, you should use GIFs for small animations, whilst larger or full-screen animations should be in MP4 format.

#### Issue

[Described detected issues supported by screenshots and code snippets, as why these things are happening to allow a client to recreate what we are see.]

#### Recommendations

[Explicit instructions on how to fix the issue supported by screenshots, code snippets with the fixed issue and mock-ups how it should look like.]

#### Reference documents

[Spreadsheets with URLs where all these issues are happening or direct links to tools that you might be using.]

### 3.3. Image Dimension

#### Overview

Because mobile devices have smaller screens, you should never use an image with the same dimensions as one tailored for desktops. On smartphones, larger images take much longer to load due to a slower Internet connection, and then need to be scaled down to fit the screen.

This uses up bandwidth for the user and makes the page display later than it should. Therefore, your images should be available in several different dimensions, as this will allow the browser to choose an image of a suitable size for each device to fit into the available space on your page.

Popular content management systems, such as WordPress, automatically create and insert these different image sizes for you when you add the image to the page, creating an effortless process.

#### Issue

[Described detected issues supported by screenshots and code snippets, as why these things are happening to allow a client to recreate what we are see.]

#### Recommendations

[Explicit instructions on how to fix the issue supported by screenshots, code snippets with the fixed issue and mock-ups how it should look like.]

#### Reference documents

[Spreadsheets with URLs where all these issues are happening or direct links to tools that you might be using.]

### 3.4. Image Compression

#### Overview

You can further reduce the size of your images by compressing them using Lossless and Lossy techniques. Lossless will allow you to reduce the size of an image by an average of around 2030% without changing the quality of the image. Lossy compression can reduce the size of an image by up to 90% with only a small loss in quality.

Images often contain unnecessary information when they appear on your pages. Unnecessary information includes: the camera the image was taken with, the date the image was made and location it was taken in. Additionally, images may contain varieties of colours that human eyes are not able to distinguish. By reducing the range of colour on the image by removing them and any unnecessary information, you are able to substantially reduce the image size, making your pages load much faster.

#### Issue

[Described detected issues supported by screenshots and code snippets, as why these things are happening to allow a client to recreate what we are see.]

#### Recommendations

[Explicit instructions on how to fix the issue supported by screenshots, code snippets with the fixed issue and mock-ups how it should look like.]

#### Reference documents

[Spreadsheets with URLs where all these issues are happening or direct links to tools that you might be using.]

### 3.5. Deferred Image Load

#### Overview

Images are usually the biggest files that a user’s browser needs to download and display your page. However, not all images need to be downloaded right away, as those further down your pages are often not seen as a person jumps from page to page.

By deferring image load, you can decrease the time it takes for your page to download. This can be done by requesting the browser downloads only the images at the top part of your page. All other images will be automatically loaded and displayed when the user gets closer to them.

This technique will decrease the time it takes for a browser to load your pages.

#### Issue

[Described detected issues supported by screenshots and code snippets, as why these things are happening to allow a client to recreate what we are see.]

#### Recommendations

[Explicit instructions on how to fix the issue supported by screenshots, code snippets with the fixed issue and mock-ups how it should look like.]

#### Reference documents

[Spreadsheets with URLs where all these issues are happening or direct links to tools that you might be using.]

## 4. Time to Load the Next Page

#### Overview

This section lists techniques used to speed up the page load of the next page on the website that a user might go to, and a repeat visit to the same page. It explains how to reuse across all pages the code downloaded by a browser on an initial page load, and how to load in advance resources that might be needed next.

This way, the next page will load much faster than the first one.

### 4.1. Prefetching

#### Overview

Using a prefetch directive allows you to tell your user’s browser to download any documents it may require for the next page. This way, the page will load much faster, as the browser will already have some of the required files.

Therefore, it is beneficial to prefetch the largest and most critical files in order to render your pages in advance. In addition to this, when data shows that users follow a specific path on your page, you can prefetch the full page due next on this path. For example, this could be your ‘offer’ page when a person visits your homepage.

This can significantly improve the time that the page takes to load. The faster the page load, the higher chance that the user will convert.

#### Issue

[Described detected issues supported by screenshots and code snippets, as why these things are happening to allow a client to recreate what we are see.]

#### Recommendations

[Explicit instructions on how to fix the issue supported by screenshots, code snippets with the fixed issue and mock-ups how it should look like.]

#### Reference documents

[Spreadsheets with URLs where all these issues are happening or direct links to tools that you might be using.]

### 4.2. Client-Side Caching

#### Overview

Your server can tell a user’s browser to save or store a copy of images, CSS, JavaScript files or the entire page for future usage. This way, when the searcher returns to your page later, the browser does not have to download it again, as it can use a saved copy of your files stored on the computer. This can greatly improve your page load time on the repeated visit.

An expiration date, which is how long those files should be stored by the browser, should be set to 1 year for your images. On your CSS, JavaScript and PDF files, this should be between 1 week and 1 month. The exact time of how long these should be stored for depends on how frequently you make changes to your site.

#### Issue

[Described detected issues supported by screenshots and code snippets, as why these things are happening to allow a client to recreate what we are see.]

#### Recommendations

[Explicit instructions on how to fix the issue supported by screenshots, code snippets with the fixed issue and mock-ups how it should look like.]

#### Reference documents

[Spreadsheets with URLs where all these issues are happening or direct links to tools that you might be using.]

### 4.3. Query Strings

#### Overview

Your website may automatically attach query strings such as ?ver=4.6 to the end of the URLs of your files. This indicates to browsers that they should not cache those files.

By removing query strings from your resources that should be cached, you allow browsers to store those files locally, decreasing the time to open the same page on a repeat visit.

#### Issue

[Described detected issues supported by screenshots and code snippets, as why these things are happening to allow a client to recreate what we are see.]

#### Recommendations

[Explicit instructions on how to fix the issue supported by screenshots, code snippets with the fixed issue and mock-ups how it should look like.]

#### Reference documents

[Spreadsheets with URLs where all these issues are happening or direct links to tools that you might be using.]

### 4.4. Redirects

#### Overview

#### Issue

[Described detected issues supported by screenshots and code snippets, as why these things are happening to allow a client to recreate what we are see.]

#### Recommendations

[Explicit instructions on how to fix the issue supported by screenshots, code snippets with the fixed issue and mock-ups how it should look like.]

#### Reference documents

[Spreadsheets with URLs where all these issues are happening or direct links to tools that you might be using.]

### 4.5. DNS Prefetching

#### Overview

The DNS prefetching directive performs a DNS lookup, checking the IP address of a server on which a file used by your website is hosted on before it is needed for the next page. This does not speed up the page load of your first web page where the user lands, although it will help each page thereafter to load faster. This may shorten the page load time of the next page a user will open by a few hundred milliseconds.

We recommend your site set up this directive on all pages for domains that are most frequently used and where they require files to be displayed on the screen.

#### Issue

[Described detected issues supported by screenshots and code snippets, as why these things are happening to allow a client to recreate what we are see.]

#### Recommendations

[Explicit instructions on how to fix the issue supported by screenshots, code snippets with the fixed issue and mock-ups how it should look like.]

#### Reference documents

[Spreadsheets with URLs where all these issues are happening or direct links to tools that you might be using.]