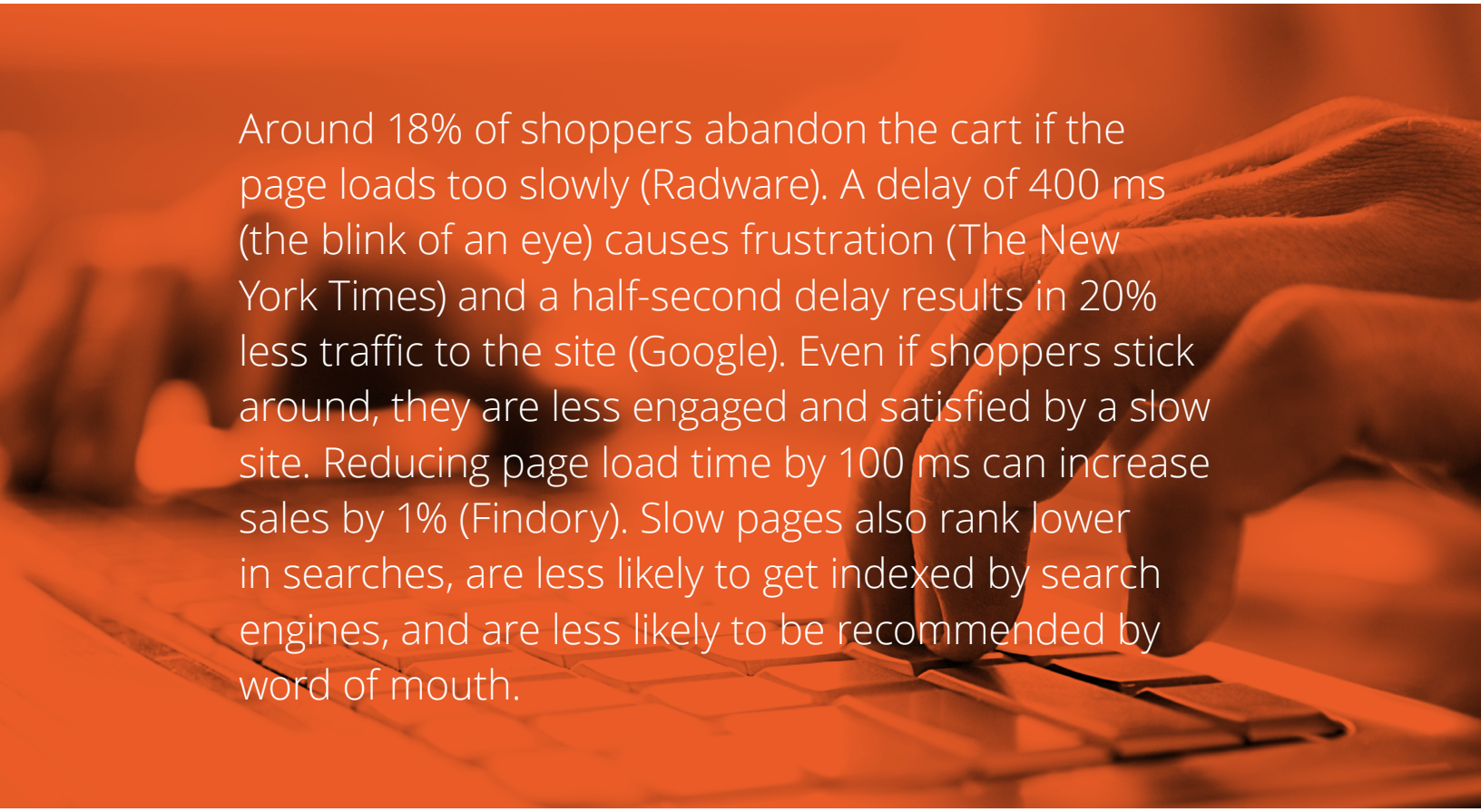
Magento
an ebay inc company

Improving Magento Front-End Performance

If your Magento website consistently loads in less than two seconds, congratulations! You already have a high-performing site. But if your site is like the vast majority of websites that take longer than two seconds to load—even for users with a fast internet connection—this article is for you.



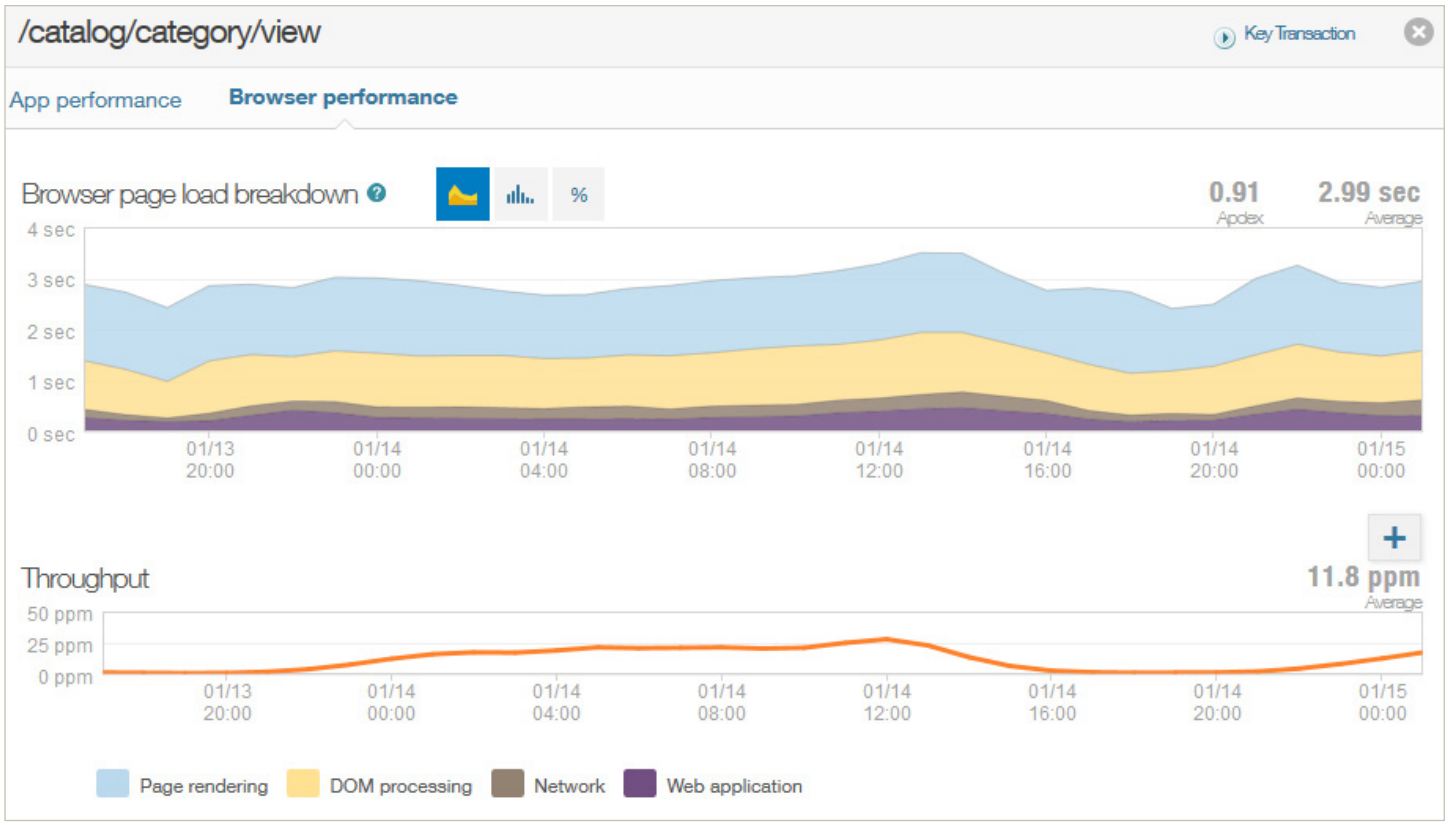
Around 18% of shoppers abandon the cart if the page loads too slowly (Radware). A delay of 400 ms (the blink of an eye) causes frustration (The New York Times) and a half-second delay results in 20% less traffic to the site (Google). Even if shoppers stick around, they are less engaged and satisfied by a slow site. Reducing page load time by 100 ms can increase sales by 1% (Findory). Slow pages also rank lower in searches, are less likely to get indexed by search engines, and are less likely to be recommended by word of mouth.

TABLE OF CONTENTS

1. WHAT IS HIGH-PERFORMANCE WEB DESIGN?

Website performance is so important that big companies of the Web (Google, Amazon etc.) are obsessed with it. What does Magento suggest?

Surprisingly, users spend a larger proportion of time waiting for a given page to load after the main HTML page has been retrieved from the server! Consider the following load diagram of a real Magento website.



The blue line is page rendering and as you see that time comprises a large percentage of page load time. To improve Magento performance, you can:

- Reduce the number of HTTP requests
- Optimize your CSS
- Optimize your JavaScript
- Optimize your images

2. REDUCING THE NUMBER OF HTTP REQUESTS

Each stylesheet, script and image in a typical HTML page requires separate round trips from the browser to the web server. The latency inherent in these HTTP requests delays the display of the page and the ability of users to interact with it. The latency can dwarf the amount of time it took to get the HTML page itself.

One effective rule of thumb for making your pages faster is to find ways to reduce the number of HTTP requests it requires. This principle is the basis for our front-end optimization techniques. The following sections discuss a few techniques for reducing HTTP requests.

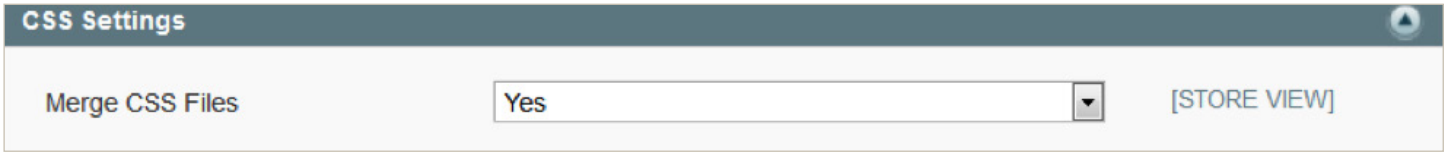
Combining Your JavaScript and CSS

Combining various CSS and JavaScript files into as few files as possible reduces the number of HTTP requests. Obviously, this is more challenging when the scripts and stylesheets vary from page to page.

Merging and Minifying Files in Magento

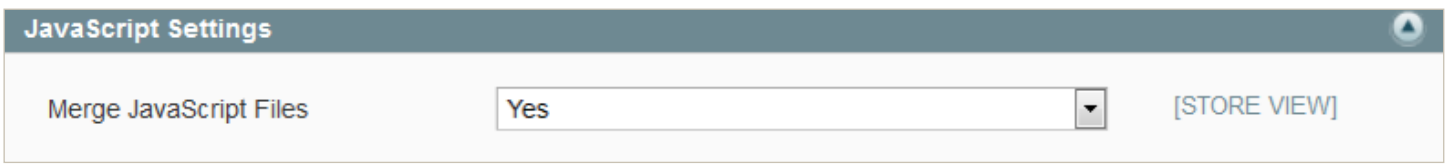
Magento enables you to merge your CSS files.

1. Log in to the Admin Panel as an administrator.
2. **Click System > Configuration > ADVANCED > Developer > CSS Settings** and select **Yes** for **Merge CSS Files**



Similarly, you can minifiy your JavaScript files as follows:

1. Under **CSS Settings**, set **Merge JavaScript** Files to **Yes**



2. Click **System** > **Cache Management.**
3. Click **Flush JavaScript/CSS Cache**.

2. REDUCING THE NUMBER OF HTTP REQUESTS (CONT'D)

To see the results of these settings, compare the following before and after.

Before

```
<script type="text/javascript" src="http://magento.loc/js/prototype/prototype.js"></script>
<script type="text/javascript" src="http://magento.loc/js/lib/ccard.js"></script>
<script type="text/javascript" src="http://magento.loc/js/prototype/validation.js"></script>
<script type="text/javascript" src="http://magento.loc/js/scriptaculous/builder.js"></script>
<script type="text/javascript" src="http://magento.loc/js/scriptaculous/effects.js"></script>
<script type="text/javascript" src="http://magento.loc/js/scriptaculous/dragdrop.js"></script>
<script type="text/javascript" src="http://magento.loc/js/scriptaculous/controls.js"></script>
<script type="text/javascript" src="http://magento.loc/js/scriptaculous/slider.js"></script>
<script type="text/javascript" src="http://magento.loc/js/varien/js.js"></script>
<script type="text/javascript" src="http://magento.loc/js/varien/form.js"></script>
<script type="text/javascript" src="http://magento.loc/js/varien/menu.js"></script>
<script type="text/javascript" src="http://magento.loc/js/mage/translate.js"></script>
<script type="text/javascript" src="http://magento.loc/js/mage/cookies.js"></script>
<script type="text/javascript" src="http://magento.loc/skin/frontend/enterprise/default/js/scripts.js"></script>
```

After

```
<script type="text/javascript" src="http://magento.loc/media/js/855c3697d9979e78ac404c4ba2c66533.js"></script>
```

3. USING CSS IMAGE SPRITES

CSS image sprites reduce the number of image requests because a sprite is a grid of images combined into a single image. Use the CSS background-image and background-position properties to display the desired image.

Using Inline Images

Inline images use the data: scheme to embed image data in an HTML page, which increases its size. Combining inline images into your (cached) stylesheets is a way to reduce HTTP requests and to avoid increasing the size of your pages. Inline images are supported across all major browsers.

Format

```
data:[<MIME-type>][;charset=<encoding>][;base64],<data>
```

The encoding is indicated by ;base64. If it is present, data is base64-encoded. Without it, the data (as a sequence of octets) is represented using ASCII encoding for octets inside the range of safe URL characters and using the standard hex encoding of URLs for octets outside that range. If <MIME-type> is omitted, it defaults to text/plain;charset=US-ASCII. (The type can be omitted but the charset must be supplied.)

Some browsers (Chrome, Opera, Safari, Firefox) accept a non-standard ordering if both ;base64 and ;charset are supplied, while Internet Explorer requires that the charset's specification must precede the base64 token.

Preventing 404s

HTTP requests are expensive so making an HTTP request and getting a useless response (such as 404 Not Found) is totally unnecessary and slows down the user experience with no benefit to the user.

4. OPTIMIZING YOUR CSS

Optimizing your CSS code is very important if you're running a large site. CSS has a tendency to get cluttered just like HTML. Here are some good rules to apply:

- No more than one external CSS stylesheet
- Small CSS in style tags for above-the-fold content
- No @import calls
- No inline CSS

Placing Your CSS in the <head> Block

Placing your CSS in the page's <head> block means that a page loads one item at time--first the title, then the logo at the top, navigation, and so on. This in turn provides sequential loading of the page and improves the site's overall user experience.

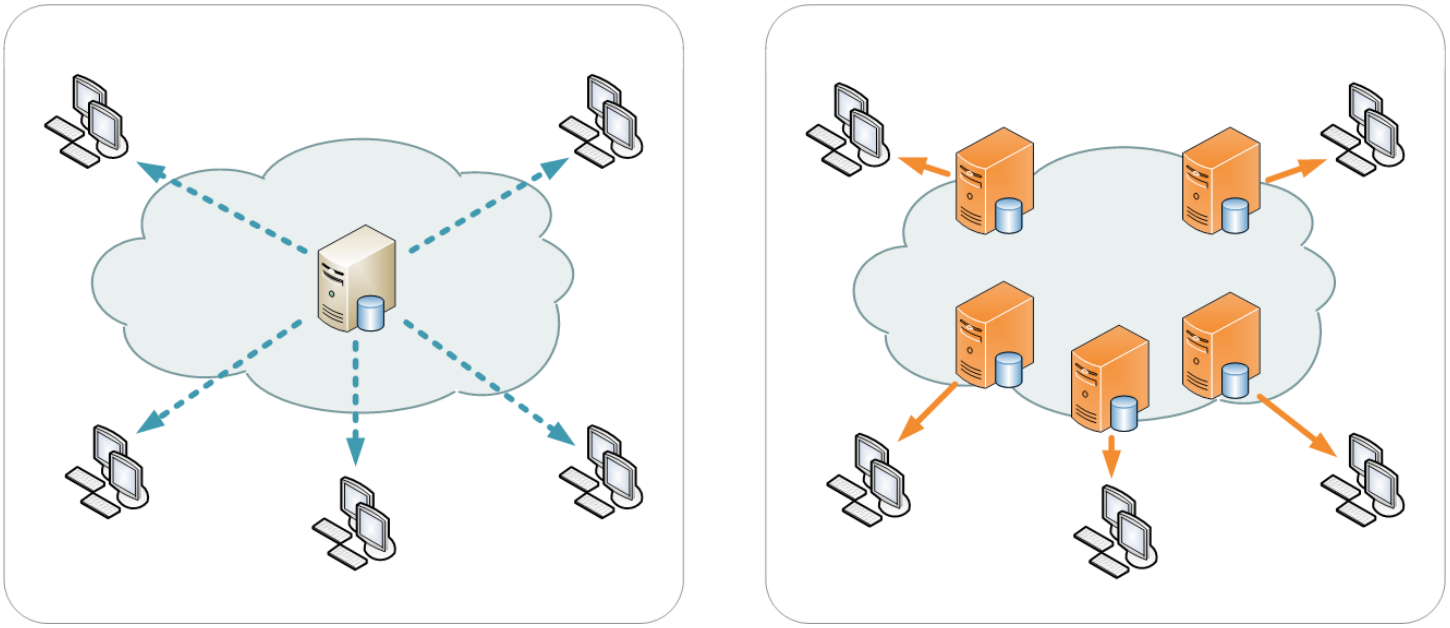
If you put the CSS files at the bottom of the page, it does not allow multiple browsers to render the page gradually. This is because the browser does not want to redraw elements, which after loading the page, might change the style.

5. OPTIMIZING YOUR JAVASCRIPT

Using a Content Delivery Network (CDN)

The user's proximity to your web server has an impact on response times. Deploying your content across multiple, geographically dispersed servers makes your pages load faster. But where should you start?

As a first step to implementing geographically dispersed content, don't attempt to redesign your web application to work in a distributed architecture. Depending on the application, changing the architecture could include daunting tasks such as synchronizing session states and replicating database transactions across servers. Attempts to reduce the distance between users and your content could be delayed by, or never pass this application architecture step.



Request handling shown without a CDN and with a CDN

Remember that 80-90% of the end-user response time is spent downloading all the components in the page: images, stylesheets, scripts, Flash, and so on. This is the Performance Golden Rule. Rather than starting with the difficult task of redesigning your application architecture, it's better to first disperse your static content. This not only achieves a bigger reduction in response times, but it's easier thanks to content delivery networks (CDNs).

A CDN is a collection of web servers distributed across multiple locations to deliver content more quickly to users. The server selected for delivering content to a specific user is typically based on a measure of network proximity. For example, the server with the fewest network hops or the server with the fastest response time is chosen.

Switching to a CDN is a relatively easy code change that will dramatically improve the speed of your web site.

5. OPTIMIZING YOUR JAVASCRIPT (CONT'D)

Best Practices for Tracking Pixels

Here we'll explain script blocking and discuss how to safely set up a tracking script (or any external script, such as Google Analytics or Quantcast) to not block page loads or other JavaScript handlers on your site.

When a browser renders a web page, it reads the HTML from top to bottom. When it encounters a <script> tag—maybe in the <head>, maybe in the <body>—it stops rendering. In other words, nothing else displays until the script has finished downloading, parsing and executing.

Later scripts get blocked because a script can modify the page, add additional scripts, or add new variables to the window object. Some modern browsers download multiple scripts in parallel (for example, Internet Explorer 8 and later or Safari 4 and later) and future versions of browsers are expected to do this too.

You can add as many tracking scripts as you want to your web page, such as Google Analytics or Quantcast. Tracking scripts, which show usage patterns, quantify how many people use your site and give you some characteristics about them.

When you add a tracking script to your site, you're adding an additional script to your pages that will follow the same blocking rules as any other script. This can be dangerous because someone else's server is hosting that script and if your servers are working fine but something is wrong with their servers, it can affect page loads on your site.

Here is an example of the Google Analytics tracking script code. Notice there are two script blocks, the first adds the ga.js script to the page and downloads it (remember that the second script is blocked from executing until this one has finished), and the second calls the tracker function with your tracking code.

Format

```
<script type="text/javascript">
var gaJsHost = (("https:" == document.location.protocol) ? "https://ssl." : "http://www.");
document.write(unescape("%3Cscript src='" + gaJsHost + "google-analytics.com/ga.js' type='text/javascript'%3E%3C/
script%3E"));
</script>
<script type="text/javascript">
try {
var pageTracker = _gat._getTracker("_your_tracking_code_here_");
pageTracker._trackPageview();
} catch(err) {}
</script>
```

Tracking scripts are designed so you can download them once and have them cached safely in your browser for future page loads on your site and even better, for page loads on any other sites that use the same type of tracking scripts. Below are some suggestions for configuring a tracking script.

5. OPTIMIZING YOUR JAVASCRIPT (CONT'D)

1. Insert scripts at the bottom of the <body>
2. Insert scripts after a load event

   This technique is designed for JavaScript-heavy websites. The tracking script code is inserted dynamically onto the page after a load event. Here is some sample code using jQuery:

```
<script type="text/javascript">
 $(window).load(function() {
  var gaJsHost = (("https:" == document.location.protocol) ? "https://ssl." : "http://www.");
  $.ajax({
    url: gaJsHost+'google-analytics.com/ga.js', type: 'get', dataType: 'script', cache: true,
    success:function() { try { _gat._getTracker("_your_tracking_code_here_")._trackPageview(); } catch(err) {} }
  });
 })
</script>
```

3. Use asynchronous tracking codes

Dependency on External Services

The Facebook **Like** button is one of the most popular things on the web today. It's important that this page loads as quickly as possible because many people use it repeatedly during their browsing sessions, even when they don't have a Facebook account. In addition to Facebook, there are other popular social networking services like Twitter, Google Plus, Pinterest and so on. We'd like to share one approach for reducing load time for these kinds of plug-ins, which in turn will increase page load speed.

Because not all plug-ins are loaded asynchronously, they affect the availability of content on the page. We recommended combining all of the requests into a single script and executing it after the content is generated. For example:

```
<script>
    (function(a, c, f) {
        function g() {
            var d, a = c.getElementsByTagName(f)[0],
                b = function(b, e) {
                c.getElementById(e) || (d = c.createElement(f),
                                        d.src = b, d.async = !0, e && (d.id = e),
                                        a.parentNode.insertBefore(d, a))
                };
            b("//connect.facebook.net/ru_RU/all.js#xfbml=1", "facebook-jssdk");
            b("https://platform.twitter.com/widgets.js", "twitter-wjs");
            b("https://apis.google.com/js/plusone.js");
        }
        a.addEventListener ? a.addEventListener("load", g, !1) : a.attachEvent && a.attachEvent("onload", g);
    })(window, document, "script");
</script>
```

This script should be inserted at the end of the code of your site, just before the closing </body>.

5. OPTIMIZING YOUR JAVASCRIPT (CONT'D)

The code's buttons...

```
<div class="fb-like" data-href="https://developers.facebook.com/docs/plugins"
                     data-layout="standard" data-action="like" data-show-faces="true" data-share="true"></div>
<a href="https://twitter.com/twitterapi" class="twitter-follow-button"
                     data-show-count="false" data-lang="en">Follow @twitterapi</a>
<div class="g-plusone" data-annotation="inline" data-width="300"></div>
```

...can be installed where needed.

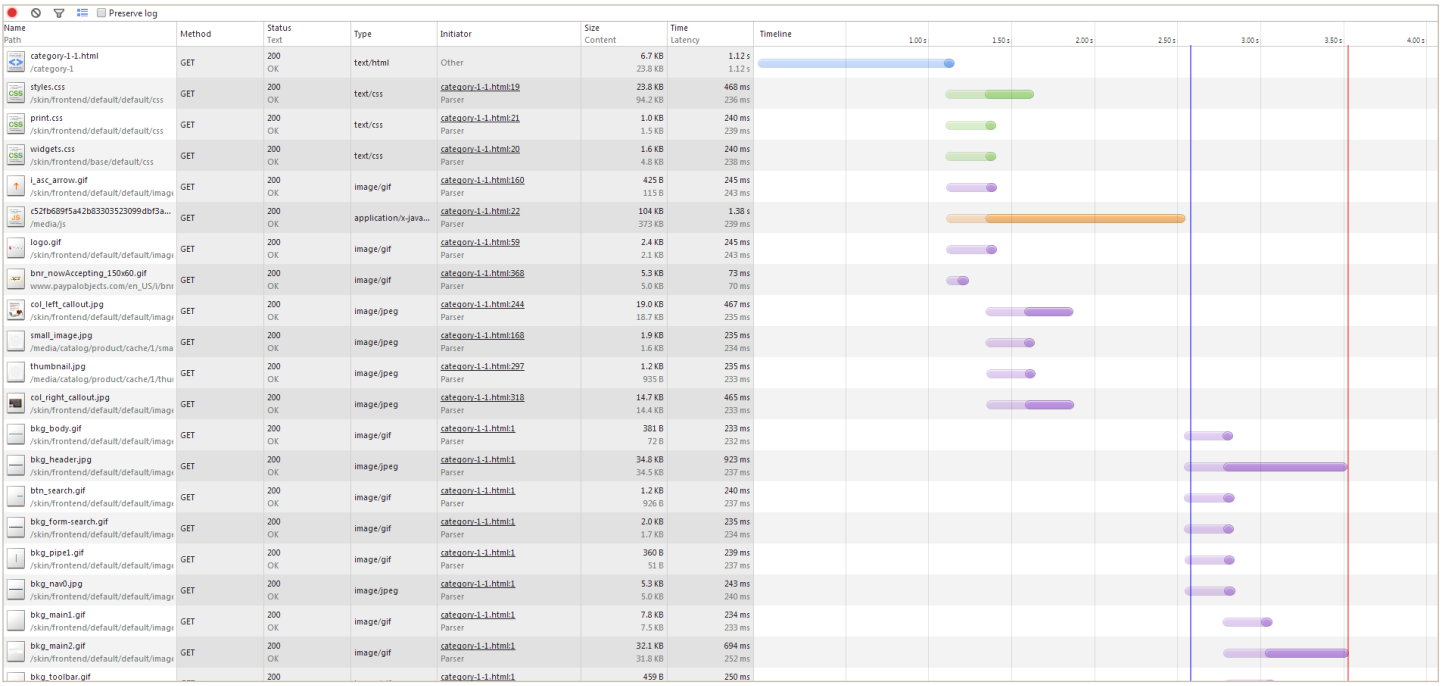
Some benefits to this approach:

- Social media buttons don't affect the speed of loading and drawing your page
- Scripts are loaded immediately after downloading more critical components of the website
- If neither the social network nor its scripts are available, your website is not affected
- All scripts and their variables are located in a single block

Deferring JavaScript

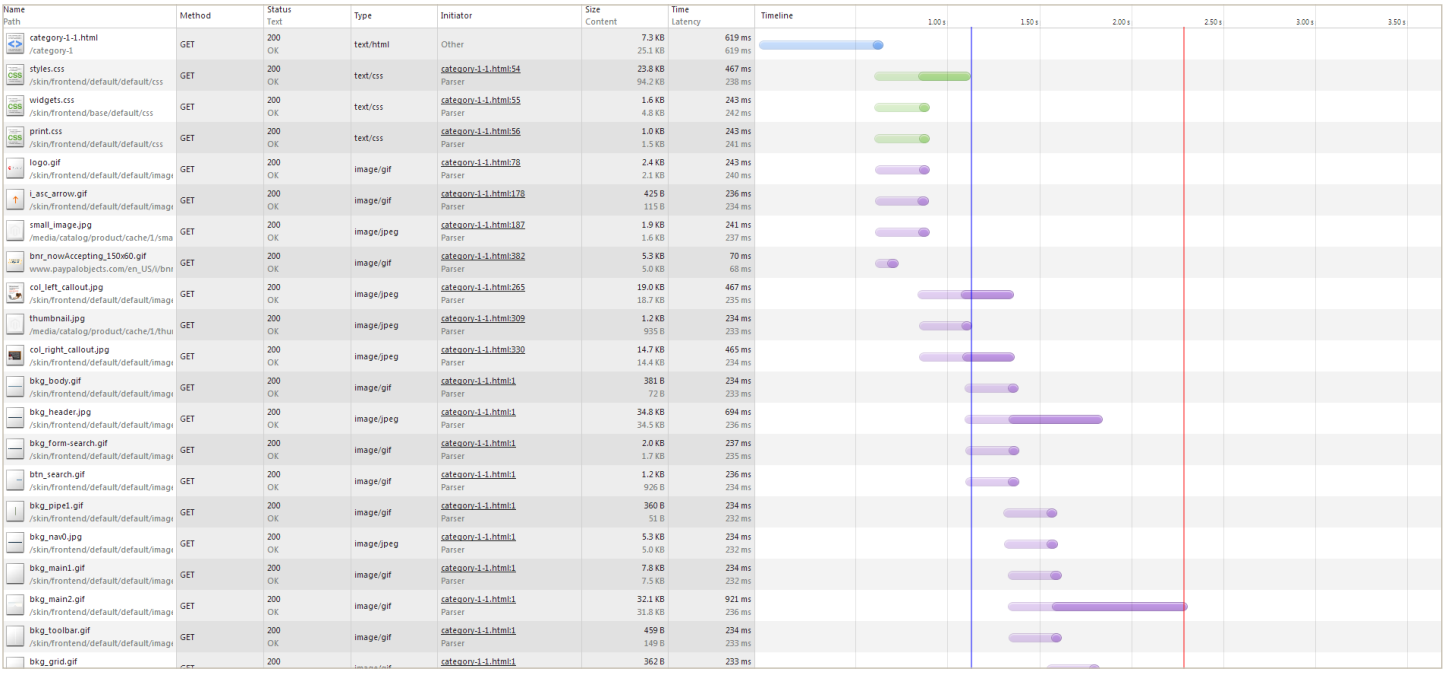
Deferring JavaScript means waiting until the page is loaded before parsing the JavaScript, which enables the page to load faster. It has no effect on the performance of your web page, but it rearranges the loading process so that JavaScript is deferred until the end. This is especially important for catalog and product pages, which usually require a lot of heavy JavaScript code.

Here's an example using browser inspectors. The first example shows how a Magento catalog page loads without deferring JavaScript:



5. OPTIMIZING YOUR JAVASCRIPT (CONT'D)

And here is the same page after deferring JavaScript:



Name / Path	Method	Status / Text	Type	Initiator	Size / Content	Time / Latency	Timeline
category-1-1.html /category-1	GET	200 OK	text/html	Other	7.3 KB 25.1 KB	619 ms 619 ms	
styles.css /skin/frontend/default/default/css	GET	200 OK	text/css	category-1-1.html:54 Parser	23.8 KB 94.2 KB	467 ms 238 ms	
widgets.css /skin/frontend/base/default/css	GET	200 OK	text/css	category-1-1.html:55 Parser	1.6 KB 4.8 KB	243 ms 242 ms	
print.css /skin/frontend/default/default/css	GET	200 OK	text/css	category-1-1.html:56 Parser	1.0 KB 1.5 KB	243 ms 241 ms	
logo.gif /skin/frontend/default/default/imag	GET	200 OK	image/gif	category-1-1.html:78 Parser	2.4 KB 2.1 KB	243 ms 240 ms	
i_asc_arrow.gif /skin/frontend/default/default/imag	GET	200 OK	image/gif	category-1-1.html:178 Parser	425 B 115 B	236 ms 234 ms	
small_image.jpg /media/catalog/product/cache/1/sma	GET	200 OK	image/jpeg	category-1-1.html:187 Parser	1.9 KB 1.6 KB	241 ms 237 ms	
bnr_nowAccepting_150x60.gif www.paypalobjects.com/en_US/i/bn	GET	200 OK	image/gif	category-1-1.html:382 Parser	5.3 KB 5.0 KB	70 ms 68 ms	
col_left_callout.jpg /skin/frontend/default/default/imag	GET	200 OK	image/jpeg	category-1-1.html:265 Parser	19.0 KB 18.7 KB	467 ms 235 ms	
thumbnail.jpg /media/catalog/product/cache/1/thu	GET	200 OK	image/jpeg	category-1-1.html:309 Parser	1.2 KB 935 B	234 ms 233 ms	
col_right_callout.jpg /skin/frontend/default/default/imag	GET	200 OK	image/jpeg	category-1-1.html:330 Parser	14.7 KB 14.4 KB	465 ms 234 ms	
bkg_body.gif /skin/frontend/default/default/imag	GET	200 OK	image/gif	category-1-1.html:1 Parser	381 B 72 B	234 ms 233 ms	
bkg_header.jpg /skin/frontend/default/default/imag	GET	200 OK	image/jpeg	category-1-1.html:1 Parser	34.8 KB 34.5 KB	694 ms 236 ms	
bkg_form-search.gif /skin/frontend/default/default/imag	GET	200 OK	image/gif	category-1-1.html:1 Parser	2.0 KB 1.7 KB	237 ms 235 ms	
btn_search.gif /skin/frontend/default/default/imag	GET	200 OK	image/gif	category-1-1.html:1 Parser	1.2 KB 926 B	236 ms 234 ms	
bkg_pipe1.gif /skin/frontend/default/default/imag	GET	200 OK	image/gif	category-1-1.html:1 Parser	360 B 51 B	234 ms 232 ms	
bkg_nav0.jpg /skin/frontend/default/default/imag	GET	200 OK	image/jpeg	category-1-1.html:1 Parser	5.3 KB 5.0 KB	234 ms 232 ms	
bkg_main1.gif /skin/frontend/default/default/imag	GET	200 OK	image/gif	category-1-1.html:1 Parser	7.8 KB 7.5 KB	234 ms 232 ms	
bkg_main2.gif /skin/frontend/default/default/imag	GET	200 OK	image/gif	category-1-1.html:1 Parser	32.1 KB 31.8 KB	921 ms 236 ms	
bkg_toolbar.gif /skin/frontend/default/default/imag	GET	200 OK	image/gif	category-1-1.html:1 Parser	459 B 149 B	234 ms 233 ms	
bkg_grid.gif		200		category-1-1.html:1	362 B	233 ms	

As you can see, deferring JavaScript considerably accelerates page load time.

5. OPTIMIZING YOUR JAVASCRIPT (CONT'D)

Example of deferring JavaScript in Magento:

```
<script type="text/javascript" >
function loadScript (){
    var callback = function(){
        if(typeof(getA) == "function" && (typeof(getacalled) == "undefined" || !getacalled))
            getA();
    };
    var url = "PATH TO THE MERGED JS FILE";
    var script = document.createElement("script");
    script.type = "text/javascript";

    if (script.readyState){  //IE
        script.onreadystatechange = function(){
            if (script.readyState == "loaded" || script.readyState == "complete"){
                script.onreadystatechange = null;
                callback();
            }
        };
    } else {  //Others
        script.onload = function(){
            callback();
        };
    }

    script.src = url;
    parentGuest = document.getElementsByTagName("body")[0];
    if (parentGuest.nextSibling) {
       parentGuest.parentNode.insertBefore(script, parentGuest.nextSibling);
    }
    else {
       parentGuest.parentNode.appendChild(script);
    }
}

if (window.addEventListener) {
    window.addEventListener("load", loadScript, false);
}
else if (window.attachEvent) {
    window.attachEvent("onload", loadScript);
}
else {
    window.onload = loadScript;
}
</script>
```

If the current page has inline JavaScript, you must move the function to the end of the page.

```
<script type="text/javascript">
var getacalled = false;
function getA(){
     ...

     YOUR SCRIPTS FOR WORK PAGE

     ....
}
</script>
```

The function is called after loading the external JavaScript.

5. OPTIMIZING YOUR JAVASCRIPT (CONT'D)

Minifing JavaScript and CSS

Minification is the practice of removing unnecessary characters from code to reduce its size and thereby improve load times. When code is minified, all comments and whitespace is removed. Minifying JavaScript improves response time performance because the size of the downloaded file is reduced. Two popular tools for minifying JavaScript code are JSMin www.crockford.com/javascript/jsmin.html) and YUI Compressor (yuilibrary.com). The YUI compressor can also minify CSS.

In addition to minifying external scripts and styles, inlined <script> and <style> blocks can also be minified. Even if you gzip your scripts and styles, minifying them will still reduce the size by 5% or more.

Note: *Keep in mind when minifying JS and CSS code is that Magento core JS code should not be minified or obfuscated!*

Obfuscating Code

Obfuscation is an alternative optimization method that can be applied to source code. It usually provides greater compression of code than minification but may result in more bugs.

---

6. OPTIMIZING IMAGES

Don't Scale Images in HTML

Don't use a bigger image than you need just because you can set the width and height in HTML. If you need

then your image (mycat.jpg) should be 100x100px rather than a scaled-down 500x500px image.

Making favicon.ico Small and Cacheable

favicon.ico stays in the root of your server. It's a necessary evil because even if you don't care about it, the browser still requests it, so it's better not to respond with a 404 Not Found. Also, because it's on the same server, cookies are sent every time it's requested. This image also interferes with the download sequence. For example, when you request extra components in the 'on load' in Internet Explorer, favicon is downloaded before these extra components.

To mitigate the drawbacks of having a favicon.ico make sure:

- It's small, preferably under 1K.
- Set the Expires header to what you feel is comfortable (because you cannot rename it if you decide to change it). You can safely set the Expires header to a few months in the future. Check the last modified date of your current favicon.ico to make an informed decision.

About Magento Expert Consulting Group (ECG)

Magento's Expert Consulting Group (ECG) helps Magento Enterprise Edition merchants and Solution Partners maximize their success. Our experts offer comprehensive analysis and best practice recommendations, from architecture planning through post-deployment.

For more expert articles from Magento ECG, visit:
magentocommerce.com/consulting/expert-articles