Team Ninja Nerds webperf-rule1

Divya Natesan & Shrikant Adhikarla 17 July 2012

Rule 1

1. Make all style sheets into one file.

   Justification(s):

   All external stylesheets are combined into a single file and referenced in index.html just once so that the number of http requests to load the styles is reduced.

2. Make all scripts into one file.

   Justification(s):

   I. All external scripts are combined into a single file and referenced in index.html just once so that the number of http requests to load the scripts is reduced.

   II. Each script inclusion using the src tag in the webpage launches a http request. We concluded that one analytics script would be good enough for tracking the website. Since, ga.js is a script from an external reference – we make sure that it is not included. Instead, we add umon2.1.js's script function(s) inside the common js file. This js file, is loaded towards the end of the body tag before before the analytic functions are acccessed in index.html.

3. Remove 404s and 410s.

   Justification(s):

   I. Two files (swfobject.js and portalinclude_v2.js) were found to not locate the resource it is referenced to and hence, were removed. This not only decreases the number of http requests but also does not change the user experience in any way.

   II.

      i. Both resources are javascripts which means that they block downloads of

elements below the <script> tag.

ii. In most browsers, the javascript resource grabbing process blocks other parallel downloads.

Such a phenomenon occurs since these scripts have the capability to change the DOM structure of the html page that is currently loading. In this particular case, a useless http request results in pausing the download of other elements of the page until the 404 response is received.

III. After executing the download of the javascript, the engine parses the code obtained trying to find something usable in it. In this case, the 404 error delays the user from viewing other important elements of the page.

IV. This process of parsing an error http response during the load of index.html wastes resources as well.

4. Remove duplicate http requests.

   Justification(s):

   There were two GET requests to the same javascript file swfobject.js (one of it was a 404, since the resource location was different). The 404 duplicate http request was removed since that page was already loading the intended <script> functions from the valid http request.

5. Make sprites.

   Justification(s):

   I. Instead of creating overheads by several http requests for each image that is being used in the page, the images are categorized into: repeatable (in the x-direction), repeatable (in the y-direction), repeatable (in both directions) and non-repeatable images.

II. Only static images (images that do not change with each page view or frequent enough) that belong to the website's domain are considered for this purpose because a single sprite image is pre-generated and used with parameters like background-position, width and height to display the part of the big image relevant to that particular element. The three images shown at the end of index.html are promo-s (can be changed frequently) and hence, excluded. Other images that are excluded from being part of the sprite are get_flash_player.gif and privacy_certified-2008.gif: this is because these images are laoded from a third party website and the ubi.com domain cannot take responsibility to update their sprite when the image (logo change etcetera) is initiated by the firm running those websites like adobe.

III. The padding between two images within the sprite is chosen in such a way that it is optimum.

    i. The horizontal offset should be large enough to avoid the background-repeat bug from Safari.

    ii. The vertical offset was not chosen to be zero, because that would interfere with the corresponding images being shown on the screen when the end-user increases the font.

    iii. Lesser padding will in turn, decrease the file size of the image allowing the browser to decompress and decode the image faster (mobile-friendly method).

IV. Different sprite images are created based on the repeat feature used by that image in the html element. For instance, all x-repeatable elements are aligned vertically and made symmetrical with respect to the width and vice-versa is done for y-repeatable elements.

Non-repeatable elements can be directly fed into the sprite generator.

V. The non-repeating foreground images are also encompassed in the sprite files. The img tag in the index.html is replaced by the specification of the image as a background. This modification brought us to add an onclick event to the tag with the image so instead of an image link originally present in the file. We ensured that all intial properties of the website was preserved. For instance, the href tag shows up a hand shaped cursor on hover. This property was added to the respective element in the new stylesheet element description.

Other Optimization Tweaks Applied

1. Put style sheets on top.

   Justification(s):

   I. Putting stylesheets to the top, makes the page appear to be faster because of progressive loading (especially for end-users with slow connection or pages with a lot of content).

   II. This mechanism also enhances the user experience because different elements of the page slowly appear indicating the progress level of the load.

   III. So why not put stylesheets at the bottom?

      i. Some browsers like IE do not render the elements until the CSS is parsed because it might have to redraw the elements if otherwise.

      ii. There is also a chance that the elements are unstyled while presented to the user during progressive load, which might not be appreciated by the user.

2. Put scripts towards the bottom.

   Justification(s):

I. As mentioned above, scripts block parallel downloads (even those served from different host names to index.html). This mechanism allows the user to view the elements on the page progressively without pausing for a long time for the <javascript> to load.

II. It was also taken care that, these javascript functions were not referenced in index.html before the script source is included to avoid scoping issues.

3. Minification of CSS and JS.

   Justification(s):

   Minification removes comments and unnecessary characters. This improves the page speed because the file size of the above mentioned components is reduced.

4. Gzipping CSS and JS.

   Justification(s):

   This improves the page speed because the file size of the above mentioned components is reduced.

5. Make favicon.

   Justification(s):

   I. Most browsers send a request to fetch the favicon, by default. Since our website did not have one, we included an icon in the project to avoid bad requests. As long as there is a http request-response interchange, the icon (of small size) can be retrieved so that the end-user can use it for bookmarking purposes.

   II. The favicon image chosen had to be small, unlike the logo itself because it interferes with the downloads of other components during page load.

   III. When a page is requested multiple times, the favicon is retrieved from the server as a

cookie.

2. Compress sprite

   Justification(s):

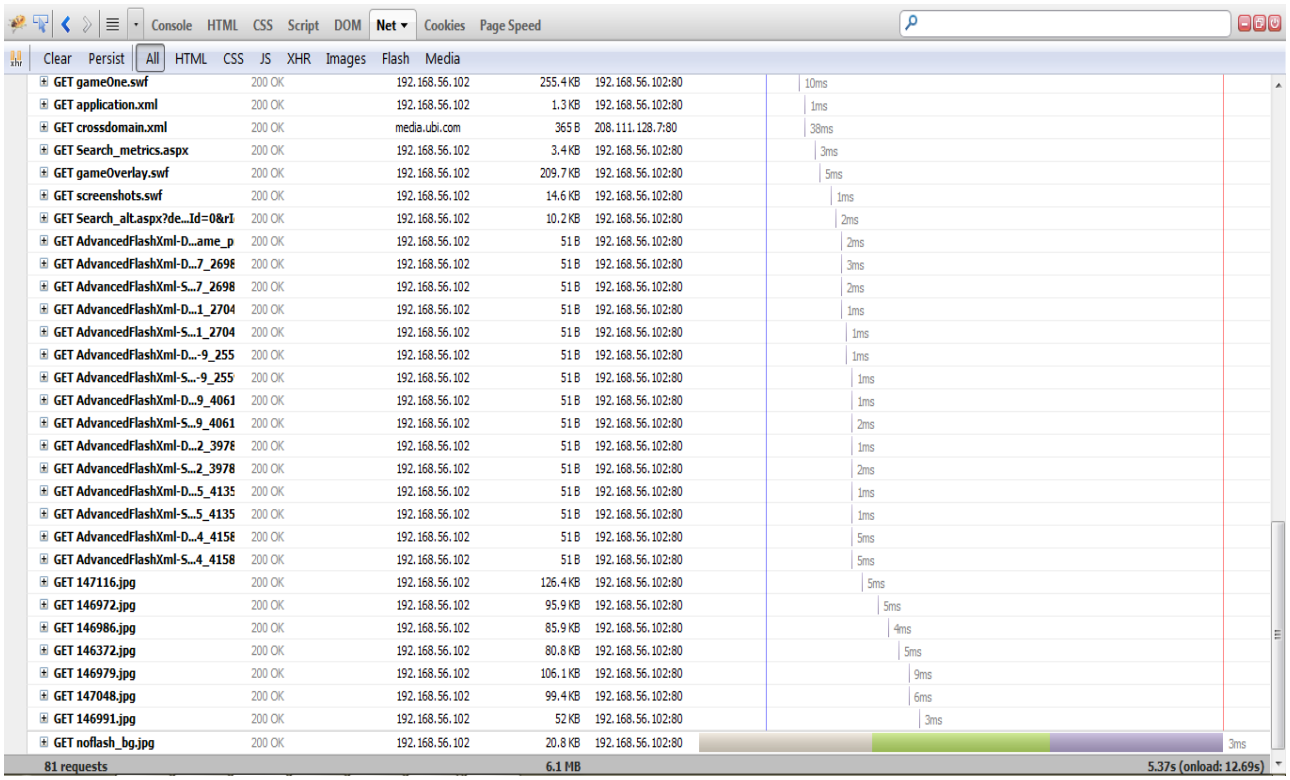
   I. The sprite image obtained is further compressed to reduce the file size.

   II. PNG format was chosen because it allows lossless compression.

All the above results were determined and verified with the help of external tools - Google Page Speed, YSlow, Google Developer Tools, Sprite Generator Tool, Firebug, YUI Compressor etcetera.

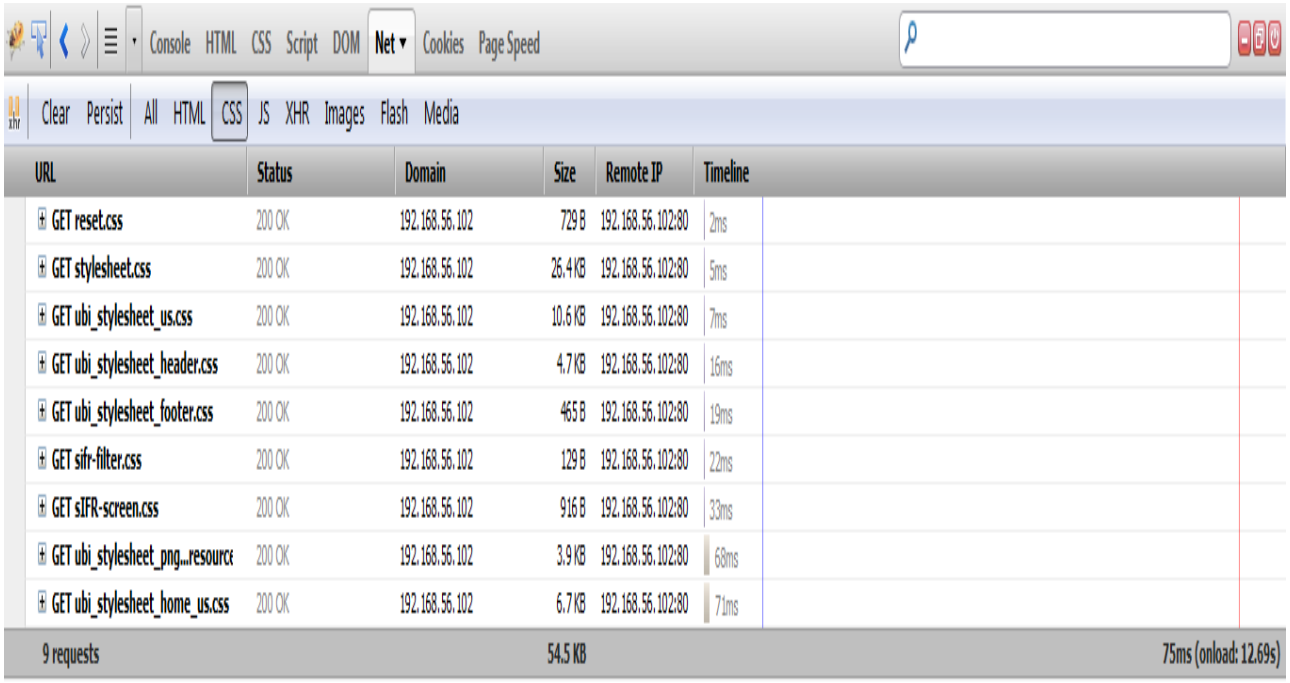
Statistics

1. *Below is a snapshot that shows the initial count of total http requests for the webpage to load* (the cache was cleared before this screenshot was captured):

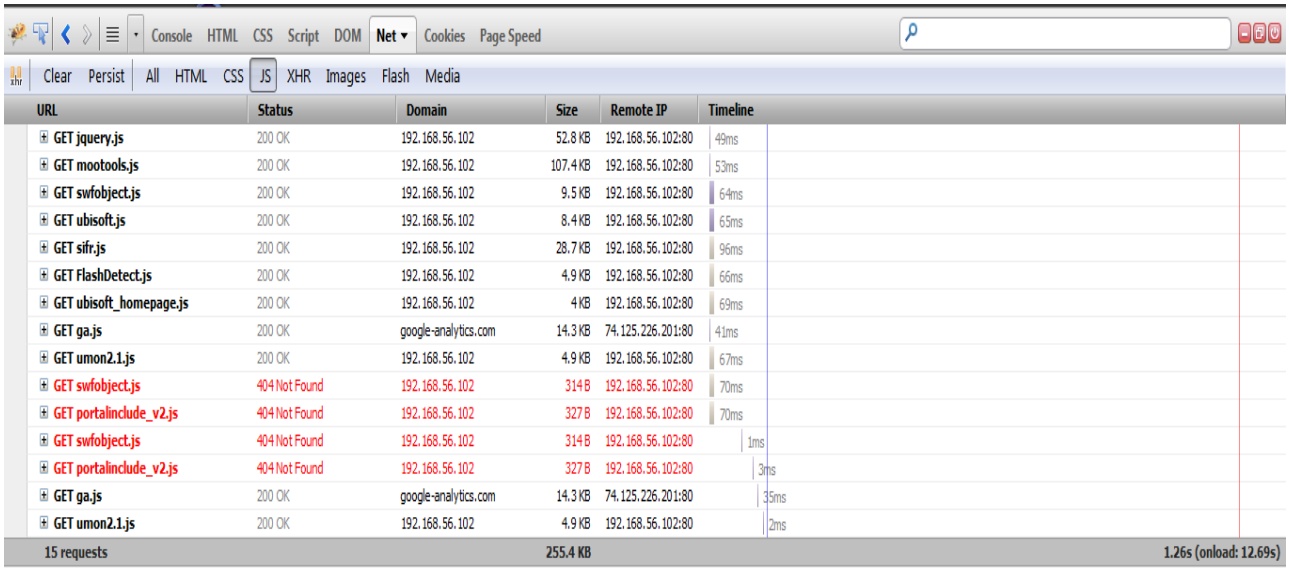


---

2. *Below is a snapshot that shows the initial count of total http requests for retrieval of stylesheets* (the cache was cleared before this screenshot was captured):

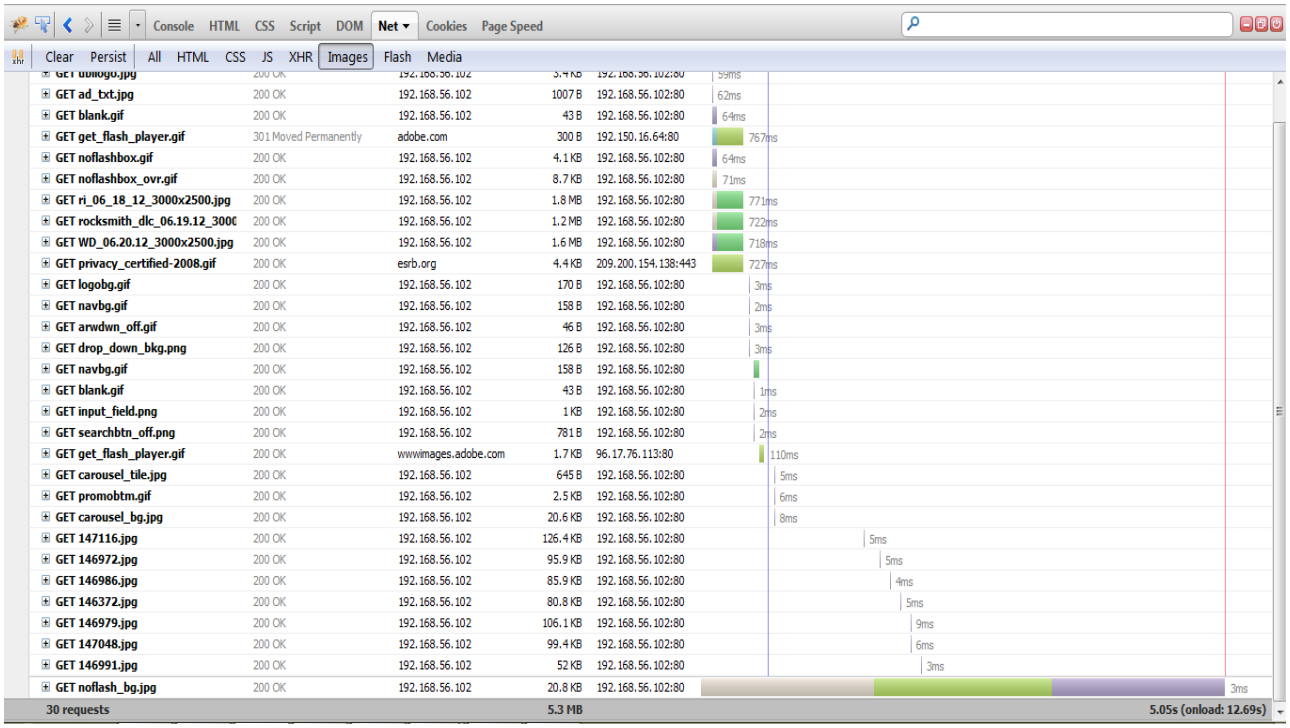


3. *Below is a snapshot that shows the initial count of total http requests for retrieval of scripts* (the cache was cleared before this screenshot was captured):

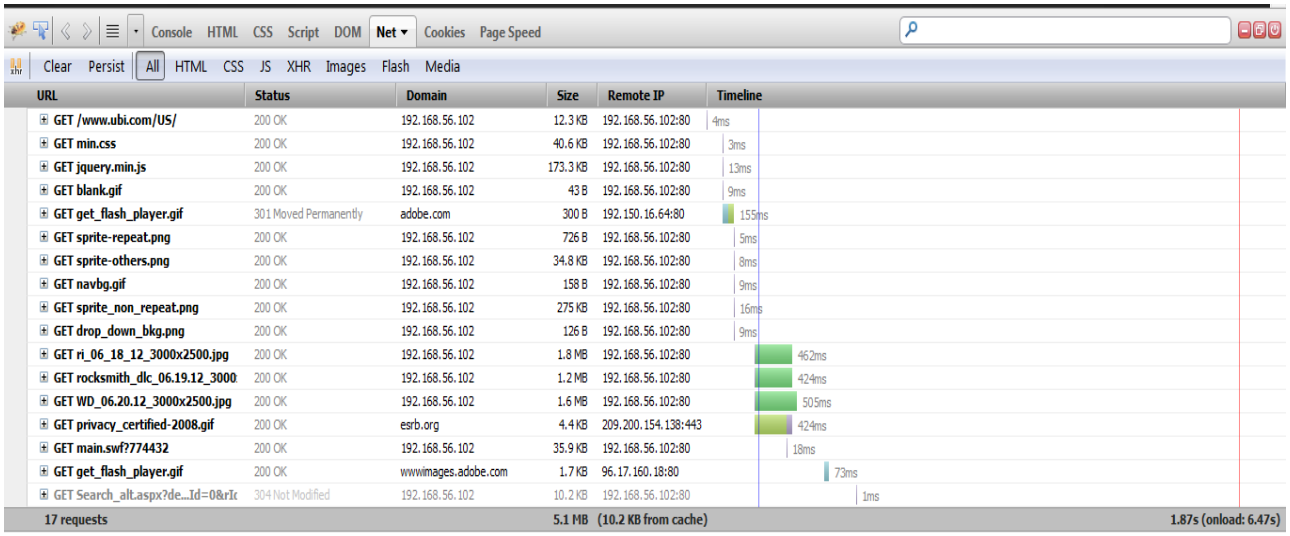


---

4. *Below is a snapshot that shows the initial count of http requests for retrieval of images* (the cache was cleared before this screenshot was captured):

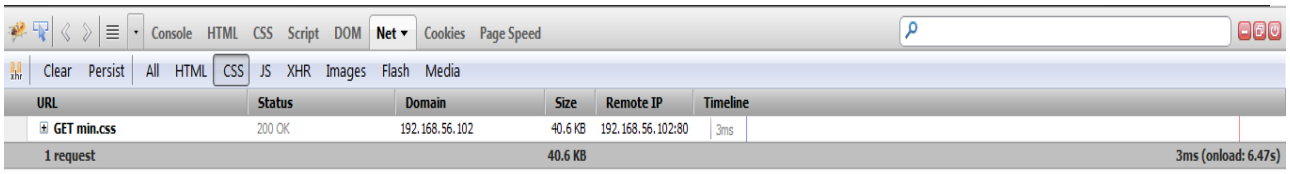


5. *Below is a snapshot that shows the current count of total http requests* (the cache was cleared before this screenshot was captured):

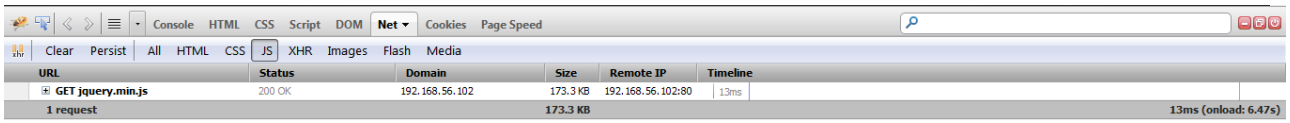


---

6. *Below is a snapshot that shows the current count of http requests for retrieval of stylesheets* (the cache was cleared before this screenshot was captured):



7. *Below is a snapshot that shows the current count of http requests for retrieval of scripts* (the cache was cleared before this screenshot was captured):



8. *Below is a snapshot that shows the current count of http requests for retrieval of images* (the cache was cleared before this screenshot was captured):