March 06, 2017

Accelerated Mobile Pages Advertisement and Cascading Style Sheet Merging

Richard Rapp
Justin Lewis

Follow this and additional works at: http://www.tdcommons.org/dpubs_series

Recommended Citation
Rapp, Richard and Lewis, Justin, "Accelerated Mobile Pages Advertisement and Cascading Style Sheet Merging", Technical Disclosure Commons, (March 06, 2017)
http://www.tdcommons.org/dpubs_series/414

This work is licensed under a Creative Commons Attribution 4.0 License.
This Article is brought to you for free and open access by Technical Disclosure Commons. It has been accepted for inclusion in Defensive Publications Series by an authorized administrator of Technical Disclosure Commons.
ACCELERATED MOBILE PAGES ADVERTISEMENT AND CASCADING STYLE SHEET MERGING

Accelerated mobile pages (AMP) offer a better user experience by improving load times for pages linked to from search results by sending a version of those pages, optimized for mobile devices, along with the search results. This is particularly useful in places where latency may be high, such as in emerging markets with poor cell infrastructure and less capable hardware. While this approach has been successful at reducing load times, one of the challenges is incorporating ads within the AMP.

Traditionally, ads are served in inline frames (also known as “iframes”) embedded within a web page. This allowed the ads to be sandboxed from the rest of the page’s content. Once the page loaded, the iframes would attempt to load ads within them. However, this goes against the general spirit of AMP, which attempts to have the page entirely loaded before the user attempts to display it. The traditional iframe approach would mean that the page would display instantly, but ads on that page could take several seconds to appear, providing a poor user experience and simultaneously lowering the effectiveness of those ads.

One approach that can be used to solve this issue is to embed the ads directly into the page without an iframe. Attempting to eliminate iframes overall is a high order goal that would greatly improve loading times, while introducing a few challenges. This would allow the ads to load as if they were a natural part of the page itself. This solution fixes the initial issue, but introduces others. By embedding the ad content directly into the page, the sandboxing mentioned above is eliminated. One of the potential problems that would arise from the elimination of the sandboxing is the potential for cascading style sheets (CSS) name collisions.
CSS is used to provide formatting and styles to elements on a web page. This is typically done by defining CSS classes, each with a different name, that are applied to the page they are formatting. CSS is ubiquitous on modern web pages, and also heavily used to format ads creatives. When loaded in an iframe, the CSS of an ad only applies to that iframe, and the CSS of the page the iframe is in does not apply to the contents of the iframe. When the iframe is eliminated, CSS used in both the ad creative and the content page would apply to the entire resulting document. In some cases, this would not cause issues. However, there is a strong potential for overlap. Previously, when ads were directly inserted into the page, they were first party ads, or at least added directly by the publisher, so conflicts would be removed manually. However, since in this case the content page and the ad creative are built by entirely different entities (a publisher and an advertiser) who are unaware exactly what ad will be on their page or what page their ad will serve on, a common name for something (e.g., “TitleText”) could be used as the name for a CSS class in both the content document and the ad creative. Even if the content page and an ad within it do not conflict, it is even more likely that a two separate ads within the same content page would share a CSS class name, also resulting in a collision. Attempting to apply both CSS classes to the resulting AMP document would have unpredictable results that would at least cause the page to display incorrectly, and at worst could cause the page to be completely unusable. Furthermore, since ads are usually inserted as the result of an auction, the pairing of an ad to a content page is not consistent, which could prove extremely difficult to debug.

The present technique includes a mechanism for cloaking CSS rules in accelerated mobile pages so that they can be embedded within the web page without interfering with the page’s layout and without using iframes to separate the elements. With the present technique the
display ads shown on the page will be able to be embedded within the page contents itself. Additionally instead of simply using a full image the AMP ads can be composed of CSS markup so that they are significantly smaller and have interactive features (such as clickable and non/clickable regions) or scaling properties. The present technique reduces not only latency and network traffic but also the total size of the embedded page. Reducing the size of the embedded page is especially useful because in the AMP protocol several pages are sent down to the client before the user chooses to click on them. Further since the contents of the page are mostly text the images tend to take up most of the pages memory size. The present technique is different from standard CSS cloaking in that it prevents collision between CSS rules from multiple sources as opposed to simply obfuscating the CSS. In fact, in some implementations of the present technique, the CSS may not be obfuscated at all.

The present technique can prevent CSS collisions from occurring even when several documents are merged together (which happens to be the case when a single AMP page includes multiple display ads). With the present technique, multiple documents are merged so that a single new document is produced with the combined page’s contents. This is different from how ads are embedded on web pages today where they use iframes (or similar) to keep the individual elements sandboxed.

The present technique includes parsing a host AMP document to identify a collection of pre-existing CSS rules and a collection of other documents that should be merged into it. The collection of other documents can be identified by specific tags within the hypertext markup language (HTML). There are two key ways that these tags are used: they can directly identify the other document that should be included and where it should be included or they can indirectly identify another document, such is the case with the tag which only says to include an
advertisement with no specification as to which advertisement. Once the document is parsed
then the list of CSS rules are put into a hash-map and the same process repeats for each of the
other discovered documents (using the same hash-map).

Every time a CSS rule is added to the hash-map, the rule is rekeyed based on the contents
of the hash-map and the document from which it surfaced. There are a few variations for how
this renaming process can work. In one implementation each of the CSS elements are renamed
so that their name includes a local identifier for the document as well as the original name. So for
example the “video-player” class in the “host” document might be renamed to “host--video-
player”. More optimized server side implementations of this process may rename the CSS
elements so that they are shorter. This optimization will significantly reduce the size of the page
being sent to the user device. In this variation every new CSS rule is given an incremental name.
CSS rules with names from different documents also have incremental names so that the name
doesn’t match the name in the other document.

After the CSS names have been added to a hash-map (keyed by the name and document
ID) the document containing them is processed so that all of the original names are replaced with
new CSS names. Finally after each of the documents has been modified they are merged
together. Usually this means inserting one document directly into the tag of the html of another
document.

There are some interesting exceptions that have to be followed. For example some tags
are special constants used ubiquitously across all web pages (such as div, span, etc.) These tags
can be ignored as long as CSS rules don’t create references to those tags as independent assets,
or in the case that they do new classes can be added to the document and to specific instances of
these tags to differentiate between the tags in each document.
This process can be done on either the client or the server. In some optimizations this process can be done by both the client and the server to achieve the lowest possible latency and highest range of dynamic ad selection.

Figure 1

Figure 1 shows HTML tags and CSS associated with an AMP host page and an advertisement page. As discussed above, the CSS rule for “video-player” is used by both the host page and the advertisement page. This could lead to collisions, where the “max-width” rule associated with either the host page or the advertisement page would be over-written. As a result, the host page or the advertisement page could be rendered incorrectly.
Figure 2 shows merged host and advertisement pages using the techniques discussed above. In particular, names of CSS elements (e.g., “video-player”) are renamed to reflect the document that these CSS elements originated from. For example, the CSS element “video-player” in the host page, a first ad page, and a second ad page is renamed “host--video-player,” “ad1--video-player,” and “ad2--video-player,” respectively. As a result, there is no collision between the CSS elements.
Figure 3 shows an example in which a host and ad pages are merged in a manner discussed above. In particular, the CSS element names have been abbreviated and shortened, resulting in a reduced page size. Even though the CSS element names are abbreviated, the names themselves are still unique between the documents, and do not affect the rendering of the merged page.
Abstract

This document describes a technique for cloaking cascading style sheet rules in accelerated mobile pages so that they can be embedded within a web page without interfering with the page’s layout and without using inline frames to separate rules. With the present technique display advertisements shown on the accelerated mobile pages will be able to be embedded within a host page contents itself without causing any conflicts due to same rule names used on a host page and an advertisement page. Additionally instead of simply using a full image the advertisement accelerated mobile pages can be composed of cascading style sheet markup so that they are significantly smaller and have interactive features (such as clickable and non/clickable regions) or scaling properties. The present technique reduces not only latency and network traffic but also the total size of the accelerated mobile pages.