**Service Worker Strategy for Digital Ads**

Cashing frequently used HTML5 assets reduces network usage and allows digital ads to render faster. Even small assets such as a .5meg image requested every time the ad displays results in over 20Gigs of data usage per month. This document describes a service worker implementation for on demand content changes while still using a local first content access strategy.

The proposed service worker model is based on dynamically creating page elements (img or video) and dynamically defining asset names. This enables us to change assets in real time. The ramifications of this method are:

1. Dynamically named assets will need unique names as service workers cache each named asset and only update the cache when a new asset name is defined  
   .
2. Even though the asset will be retrieved from cache on subsequent plays a network connection is still necessary to retrieve the name of the dynamically named asset.  
   .
3. When a network connection is not present the page will display a fallback graphic from cache that must be defined on the page.

In our supplied examples we offer two methods of retrieving asset names:

1. Ajax 🡪 php script
2. Globals.js

In both cases network access is needed to retrieve the asset names. Lack of network access will display the fallback image and produce an error in the console.

**Service Worker Detail with Globals.js**  
The service worker implementation consists of the following assets

* *swIndex.html* - the main page, contains script testing for service worker compatibility and then to register the service worker. swIndex.html consists of three main parts:
  1. *style definition* for the background image to be displayed when offline
  2. *script to dynamically create and add the primary image*. Dynamically creating this image allows us to load the image name from js/globals.js
  3. *script to register the service worker*
* swImage.js – installs the service worker and defines assets to be cached
* js (Folder)
  + globals.js – contains global variable names of the dynamic digital assets that need to be cached
* Images (Folder) – contains referenced images

**Service Worker Detail with Ajax**To utilize Ajax the process is the same as with Globals.js with two exceptions:

1. Globals.js is omitted and replaced with Ajax
2. A PHP or similar server side script is required to supply the dynamically defined asset names

Please refer to the sample files for sample Ajax and callback hierarchy.