Event-Based Implementation

Last Modified on 02/27/2023 6:12 pm EST

Event-Based Implementation

Introduction

Introduction

If you are looking to call the ads outside of a normal page load, you will want to use our event-based ads. There are many examples of use cases for this, including:

- A user plays a game on the site and after it ends, the results appear in a popup window. While loading the content in that popup, you can also load an ad there as well.
- Articles/pages that use an infinite scroll event to start to show more content as the user scrolls.
- On a site with videos and clips, when a user pauses the video or takes another action while on the same page, you could call ads using this method.
- If you want to take advantage of Lazy Loading ads, you can use this method to load the ads as the events trigger. Most lazy loading functionality is built behind the scroll distance of the user, so for example if the user scrolls within 500 pixels of where the ad would be loading, you could set an event there to call it ahead of time so that it's loaded when the user scrolls it in to view. Note we also offer our dynamic in-content ads that have this functionality built-in already. Talk to your account manager if you are interested in this product.
- Freestar's code does not work with prototype.js, this will need to be removed from your site to add Freestar.

Pubfig.js API

Freestar has exposed a few methods in our code for the publisher to be able to call ads around events. Using the tabs in this document we define the methods exposed and what can be passed through them for use on your website.

Freestar Queue

Freestar Queue

The Freestar queue lives on top of the googletag queue. Most publishers will be familiar with some code from the GPT library that looks like this: **googletag.cmd.push();** The queue is there so if the

Freestar library is not ready, code will get pushed into a queue so that when it is ready it can process everything. Using our standard ad tag setup we do something similar where we push the ad placements into an array and when our scripts are ready, our initCallback function fires and sends the array to the ad server.

When doing event based ad loading, this helps eliminate errors in the console where scripts fire before others are ready. Below is an example of how the Freestar queue looks.

```
freestar.queue.push(function() {
   // Do some code here
});
```

freestar.newAdSlots

freestar.newAdSlots

This method allows you to pass two arguments. The first, elements, should be formatted in an array and each placement will be set up in an Object. You can pass multiple objects through the array as you see fit. We recommend pushing each object to an array and calling newAdSlots only once per page view. This allows the ads to be sent to the ad server in a single request and will help speed up the ad serving process.

The second argument is for the channel, which is a configuration item that some publishers may need to take advantage of. See the Channel Variable section for more information on this. The channel argument is optional and does not need to be set if it's not being used.

```
freestar.queue.push(function() {
   freestar.newAdSlots(elements, channel = null)
});
```

The element object has two required properties that need to be passed through and one optional property. Take a look at the example below to see how this can be used. This is showing a call for two ads on the page. The DIV's already exist in this scenario, you can also write javascript to inject the divs into the DOM if they don't already exist on the page.

```
// A complete example showing all arguments being passed
freestar.queue.push(function() {
   freestar.newAdSlots([{
      placementName: 'myplacement_300x250_InContent',
      slotId: 'my-element-id',
      targeting: {
         foo: 'bar',
         bar: 'baz'
      }
   },{
      placementName: 'myplacement_300x250_InContent',
      slotId: 'my-element-id',
      targeting: {
         foo: 'bar',
         bar: 'baz',
         many: ['value1', 'value2']
      }
   }],
   "foobar");
}):
```

Below are the definitions of the keys in the elements object that is passed.

Key Name	Type	Required	Definition
placementName	String	true	This comes from the tag provided by Freestar. This field must match what is provided by Freestar for the ads to load. Unlike slotId's placementNames can be used more than once on a page.
slotId	String	true	This is the Div ID on the page that you are trying to load the ad into. This value must be unique and only called once on the page.
targeting	Object	false	In the targeting object, you will pass a key and a value for GAM to use for targeting. You can pass as many as needed here, and you can make an array of strings if there is more than one value needed to be passed.

The last thing passed in this method is the channel name/string. Normally this is set in the header, but if you are navigating to a new section of the site without reloading the page, you will need a way to set the channel to the proper path. More of this is explained in the Channel Variable section.

freestar.deleteAdSlots

freestar.deleteAdSlots

When an event occurs on the page without a full page refresh, you should also remove the ad from the page if it's no longer going to be visible. You can use the deleteAdSlots method to remove one or more ads on the page. You can pass the string of the slotId you want to remove or you can pass an array of those slotIds if you want to remove more than one ad. If you don't pass anything, all ads will be removed from the page. Below are some examples of how this can be used. This needs to be called after the window.freestar object and window.freestar.queue array have been defined for this to work correctly.

```
// Delete all ad slots on a page
freestar.queue.push(function() {
   freestar.deleteAdSlots();
});

// Delete one of the ad slots on the page based on the slotId
freestar.queue.push(function() {
    freestar.deleteAdSlots('ad_300x250_1');
});

// Delete two of the slots. Note to pass the strings in an Array
freestar.queue.push(function() {
    freestar.deleteAdSlots(['ad_728x90_1', 'ad_300x250_2']);
});
```

Pageview Tracker

Freestar Pageview Tracker

How to Track Virtual Page Views Through Pubfig

Pubfig version 4.6.0 +

Pubfig.js collects data values such as URL location which is then used in various tables. Inorder to properly track data sites that are using Single Page Applications (SPAs), or sites with slideshows/carousels that change URLs / URL parameters the publisher must take these new actions to assure the accuracy of the collected data.

When the location and or URL is updated, the lifecycle of the DOM and or Window does not reload the pubfig.js script. In order to address this, the publisher must invoke the freestar.trackPageview() method. This will ensure that the new URL is stored and used throughout the data collection for that page or view.

In order to fend off timing issues, the function must be invoked within the Freestar queue. e.g.:

```
// (1) new view / url state updated

// (2) upon completion of the view being mounted invocation of the trackPageview method within the queued method

freestar.queue.push(function(){
  freestar.trackPageview()
});
```

Reusing a Placement

Reusing a Placement On the Same Page

When using the newAdSlots method, you can use the same placementName over and over on a page. That means that in an infinite scroll scenario, or if you are doing any other event-based ad loading you won't need Freestar to provide you with multiple placementNames to accomplish loading more ads. However, if you choose to reuse the placementName, the reporting will not break down the ads individually. Reporting will show just the placementName, so if you called it 4 times on a page you will see all of that data combined.

In order to get reporting down to the 4 times you called the ad, you will need 4 placementNames from Freestar. **It's our recommendation that you do break the ads out so that you get the reporting down to the placement level, but for some cases, it makes sense to just reuse it.**

If you choose to reuse the same placementName on the page, then all you need to do is make the slotId and div id unique. We recommend you make these the same as the placementName and then just add an incremental number to the end making the slotid and idiv id match. You can write a quick javascript for loop to increment that number and append it each time the ad is called. If you need help, reach out to your Onboarding Specialist or Customer Service Manager.

Example Code
Note: Please use your site-specific ad-tags instead of the example below.

```
<!-- Tag ID: yoursite_incontent_leaderboard_10 -->
<div align="center" data-freestar-ad="__300x250 __970x250" id="yoursite_incontent_leaderboard_10">
<script data-cfasync="false" type="text/javascript">
 freestar.config.enabled_slots.push({
    placementName: "yoursite_incontent_leaderboard_10", slotId: "yoursite_incontent_leaderboard_10" });
 </script>

 <!-- Tag ID: yoursite_incontent_leaderboard_10_2 -->
<div align="center" data-freestar-ad="__300x250 __970x250" id="yoursite_incontent_leaderboard_10_2">
<script data-cfasync="false" type="text/javascript">
 freestar.config.enabled_slots.push({
    placementName: "yoursite_incontent_leaderboard_10", slotId: "yoursite_incontent_leaderboard_10_2" });
 </script>

  <!-- Tag ID: yoursite_incontent_leaderboard_10_3 -->
<div align="center" data-freestar-ad="__300x250 __970x250" id="yoursite_incontent_leaderboard_10_3">
<script data-cfasync="false" type="text/javascript">
 freestar.config.enabled_slots.push({
    placementName: "yoursite_incontent_leaderboard_10", slotId: "yoursite_incontent_leaderboard_10_3" });
 </script>

  <!-- Tag ID: yoursite_incontent_leaderboard_10_4 -->
<div align="center" data-freestar-ad="__300x250 __970x250" id="yoursite_incontent_leaderboard_10_4">
<script data-cfasync="false" type="text/javascript">
 freestar.config.enabled_slots.push({
    placementName: "yoursite_incontent_leaderboard_10", slotId: "yoursite_incontent_leaderboard_10_4" });
 </script>
```