# Prevent Banners from Blocking

**On This Page:**



## Prevent Banners from "Blocking".

A HOWTO guide to help prevent banners from "blocking" the loading of website content.

---

### Discussion

Web browsers traditionally loaded HTML content "synchronously" - that is, the entire HTML page was loaded first, so that the browser could lay out all the content, and then other elements (such as images) were loaded after the HTML was parsed and laid out.

However, this means that when calling external sources to add in HTML content at load time (i.e. adding some Revive Adserver tags to a page), the browser needs to fetch the HTML for that content - all such content - before the page can be loaded. If the external source takes a while to load, and/or if there are many such external sources to load, this can result in very poor page load times - the external content "blocks" the main content from being loaded.

There are a few ways that Revive Adserver can be used to prevent banners from "blocking" the loading of the main site content.

### HOWTO

#### Option 1: Use the Asynchronous or iFrame Tags

Probably the easiest way to prevent Revive Adserver banners from blocking the loading of website content is to use one of the two zone invocation tags that natively load asynchronously - the Asynchronous JS Tag and iFrame Tag.

These two zone level invocation tags are simple to use - simply generate the tag(s) for the zone(s) you want to display banners for, and put the tags into your HTML content where you want the banners to appear. That's it - the banners will be loaded asynchronously.

#### Option 2: Use the Single Page Call Tag

Technically, the Single Page Call tag is a blocking tag - it is not loaded asynchronously. However, SPC is delivered in two parts - firstly, a JavaScript script that (with a little configuration) efficiently gets *all* the banners you need for a page in one HTTP request (so, it is much more efficient than loading each banner with a separate request), and then a number of separate JavaScript scripts to place each banner's HTML content into the page.

So, while on its own, the Single Page Call tag isn't technically a non-blocking way of loading banners, you may want to consider if using it is easier than using Asynchronous JS Tags or iFrame Tags (if you have many banners per page) and/or if it is efficient enough that you don't actually need to use an asynchronous loading mechanism.

#### Option 3: Bottom Loading & CSS Positioning

The final option for non-blocking loading is, again, technically not an asynchronous mechanism. However, if you feel that the Asynchronous JS Tag or iFrame Tag types are not for you, and the Single Page Call tag isn't good enough - or you need to use one of the other zone invocation tag types for any number of other valid reasons - then putting the tag(s) at the very bottom of the page's HTML, and positioning the content via CSS, may be your best option.

Although the web browser will still show that it is loading the page until all of the banners have been loaded (hence it not being a truly asynchronous mechanism), by taking this approach, all of the "main" website content will at least be loaded first, with the banners being loaded at the end. Given that this is the whole purpose of the truly asynchronous approach, using an approximation in this way may be a viable alternative.

### Notes

Please see the troubleshooting guide on Requests Higher than Impressions. The use of asynchronous loading of banner content, while perhaps "obviously" a good thing, can potentially have unintended consequences on the number of banner impressions recorded, especially if your Revive Adserver installation is underpowered (or not tuned) to handle serve the volume of banner impressions requested of it.