Synchronous and asynchronous snippet loading

Skip Ahead

Synchronous and asynchronous loading types explained
Asynchronous loading: strengths and drawbacks

THIS ARTICLE WILL HELP YOU:

• Decide when to use synchronous or asynchronous loading in the Optimizely snippet

A/B testing scripts can be loaded in two ways:

• Synchronously, where scripts are loaded sequentially, one after another, starting with the <head> tag
• Asynchronously, where some scripts can be loaded simultaneously

Loading a script asynchronously has the advantage of not slowing the page down, but it can cause a "flicker" on the page, where the original page loads, followed shortly by the variation page.

Optimizely uses a synchronous snippet to prevent flickering and a balanced content delivery network (CDN) system to make sure the impact on the page is minimized.

Do you just want to learn how to implement the Optimizely Snippet? Go to our article on implementing the Optimizely Snippet before you read this one.

You can also learn about custom snippets in Optimizely Web or how to load variation code asynchronously within in a synchronous snippet.

If you specifically want to know about asynchronous and synchronous implementation, read on.

Synchronous and asynchronous loading types explained

A web page consists of a head and a body. Everything in the body is rendered by the browser while the head is used to
load external resources (such as scripts and style sheets) and to add meta data to the page. When the page loads in a browser, the browser starts reading the html from top to bottom. The head section has a special characteristic: normally, the browser will not show anything (a white screen) until all the external resources are fully loaded. This is called "synchronous loading."

However, you could override this functionality to make certain elements load without waiting for all external resources to load. This is known as "asynchronous loading."

Here's what a page's <head> tag would look like when all resources are loaded in a synchronous manner:

```html
<html>
  <head>
    <script src="1.js" />
    <script src="2.js" />
    <script src="3.js" />
    <link href="4.css" />
    <script src="5.js" />
  </head>
  <body>Hello World!</body>
</html>
```

When all resources in the head are synchronous, the elements will load in order, like this:

<table>
<thead>
<tr>
<th>Resources</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Script 1</td>
<td></td>
</tr>
<tr>
<td>Script 2</td>
<td></td>
</tr>
<tr>
<td>Script 3</td>
<td></td>
</tr>
<tr>
<td>Css 4</td>
<td></td>
</tr>
<tr>
<td>Script 5</td>
<td></td>
</tr>
</tbody>
</table>

Here's what a page's <head> tag would look like when all resources are loaded in an asynchronous manner:

```html
<html>
  <head>
    <script async src="1.js" />
    <script async src="2.js" />
    <script src="3.js" />
    <link href="4.css" />
    <script src="5.js" />
  </head>
  <body>Hello World!</body>
</html>
```
When a script is asynchronous, it will load simultaneously with other scripts, like this:

![Diagram showing script loading](image)

You can see in the diagram, that scripts 2 and 3 are now able to load at the same time, which speeds up the overall loading of a page. However, the browser will not wait until the snippet is done loading before displaying the elements in the body to a visitor, which can lead to "flickering" (see below).

**Asynchronous loading: strengths and drawbacks**

On the plus side, loading asynchronously will prevent any delay in pageload because the page will attempt to load all elements simultaneously, including your A/B testing scripts.

However, when an A/B testing script is loaded much later than the page, a flicker of the page can occur. This flicker happens when the original page loads, followed by the variation. This is not desirable, especially when the loading of the A/B testing script is taking more time than is acceptable.

So in the asynchronous example above, a visitor might see "Hello world" in her browser before script 2 is done loading. If the A/B testing script is not loaded before the webpage is shown, it will result in a "flicker." Flickering means that the variation will load on the page shortly after the original page displays. This will make the outcome of your experiment less reliable and could compromise your visitor experience.

In Optimizely, the snippet can be loaded synchronously and variations can be loaded asynchronously. Read more on this topic in this article: [Load Variation Code Synchronously or Asynchronously](#)