# Asynchronous vs Deferred JavaScript

*Feb 28, 2017* *javascript*, *performance*

In my article on Understanding the Critical Rendering Path, I wrote about the effect JavaScript files have on the Critical Rendering Path.

> *JavaScript is considered a "parser blocking resource". This means that the parsing of the HTML document itself is blocked by JavaScript. When the parser reaches a* ***<script>*** *tag, whether that be internal or external, it stops to fetch (if it is external) and run it.*

This behaviour can be problematic if we are loading several JavaScript files on a page, as this will interfere with the time to first paint even if the document is not actually dependent on those files.

Fortunately, the **<script>** element has two attributes, **async** and **defer**, that can give us more control over how and when external files are fetched and executed.

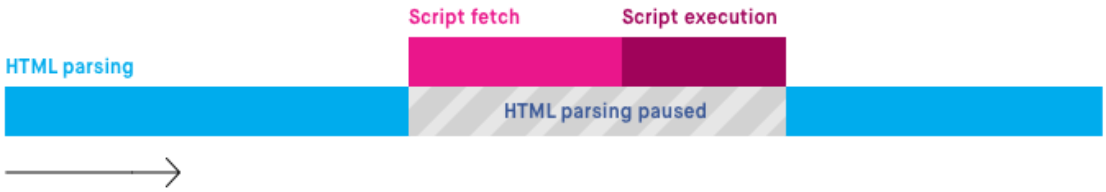
## Normal Execution

Before looking into the effect of the two attributes, we must first look at what occurs in their absence. By default, as mentioned above, JavaScript files will interrupt the parsing of the HTML document in order for them to be fetched (if not inline) and executed.

Take, for example, this script element located somewhere in the middle of the page -

```
<html>
<head> ... </head>
<body>
    ...
    <script src="script.js">
    ....
</body>
</html>
```

As the document parser goes through the page, this is what occurs -



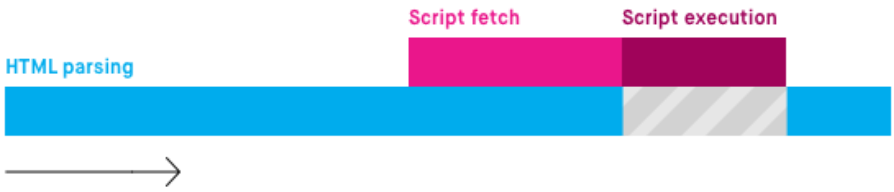

The HTML parsing is paused for the script to be fetched and executed, thereby extending the amount of time it takes to get to first paint.

## The `async` Attribute

The **async** attribute is used to indicate to the browser that the script file *can* be executed asynchronously. The HTML parser does not need to pause at the point it reaches the script tag to fetch and execute, the execution can happen whenever the script becomes ready after being fetched in parallel with the document parsing.

```
<script async src="script.js">
```

This attribute is only available for externally located script files. When an external script has this attribute, the file can be downloaded while the HTML document is still parsing. Once it has been downloaded, the parsing is paused for the script to be executed.

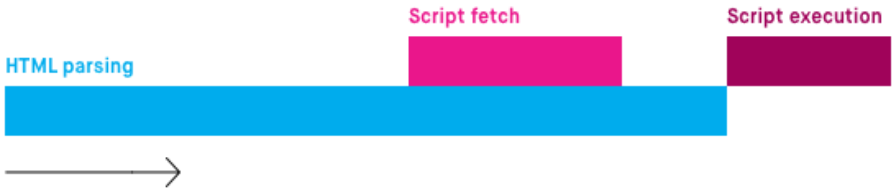



## The `defer` Attribute

The **defer** attribute tells the browser to only execute the script file once the HTML document has been fully parsed.

```
<script defer src="script.js">
```

Like an asynchronously loaded script, the file can be downloaded while the HTML document is still parsing. However, even if the file is fully downloaded long before the document is finished parsing, the script is not executed until the parsing is complete.




## Asynchronous, Deferred or Normal Execution?

So, when should we use asynchronous, deferred, or normal JavaScript execution? As always, it depends on the situation, and there are a few questions to consider.

### Where is the <script> element located?

Asynchronous and deferred execution of scripts are more important when the **<script>** element is not located at the very end of the document. HTML documents are parsed in order, from the first opening **<html>** element to it's close. If an externally sourced JavaScript file is placed right before the closing **</body>** element, it becomes much less pertinent to use an **async** or **defer** attribute. Since the parser will have finished the vast majority of the document by that point, JavaScript files don't have much parsing left to block.

### Is the script self-contained?

For script files that are not dependent on other files and/or do not have any dependencies themselves, the **async** attribute is particularly useful. Since we do not care exactly at which point the file is executed, asynchronous loading is the most suitable option.

### Does the script rely on a fully parsed DOM?

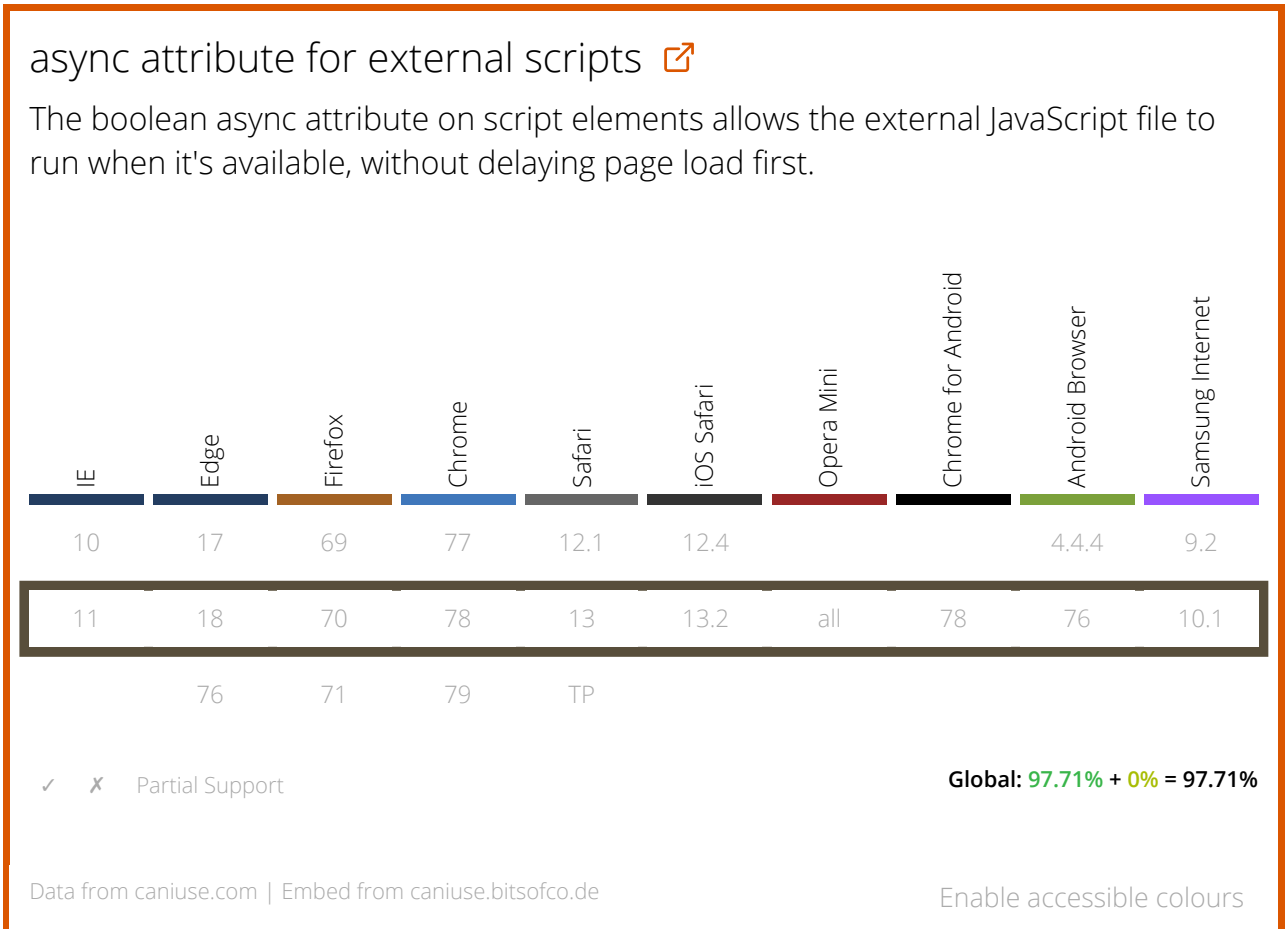
In many cases, the script file contains functionality that requires interaction with the DOM. Or, it may have a dependency on another file included on the page. In these cases, the DOM must be fully parsed before the script should be executed. Typically, such a file will be placed at the bottom of the page to ensure everything before it has been parsed. However, in situation where, for whatever reason, the file in question needs to be placed elsewhere, the **defer** attribute can be used.

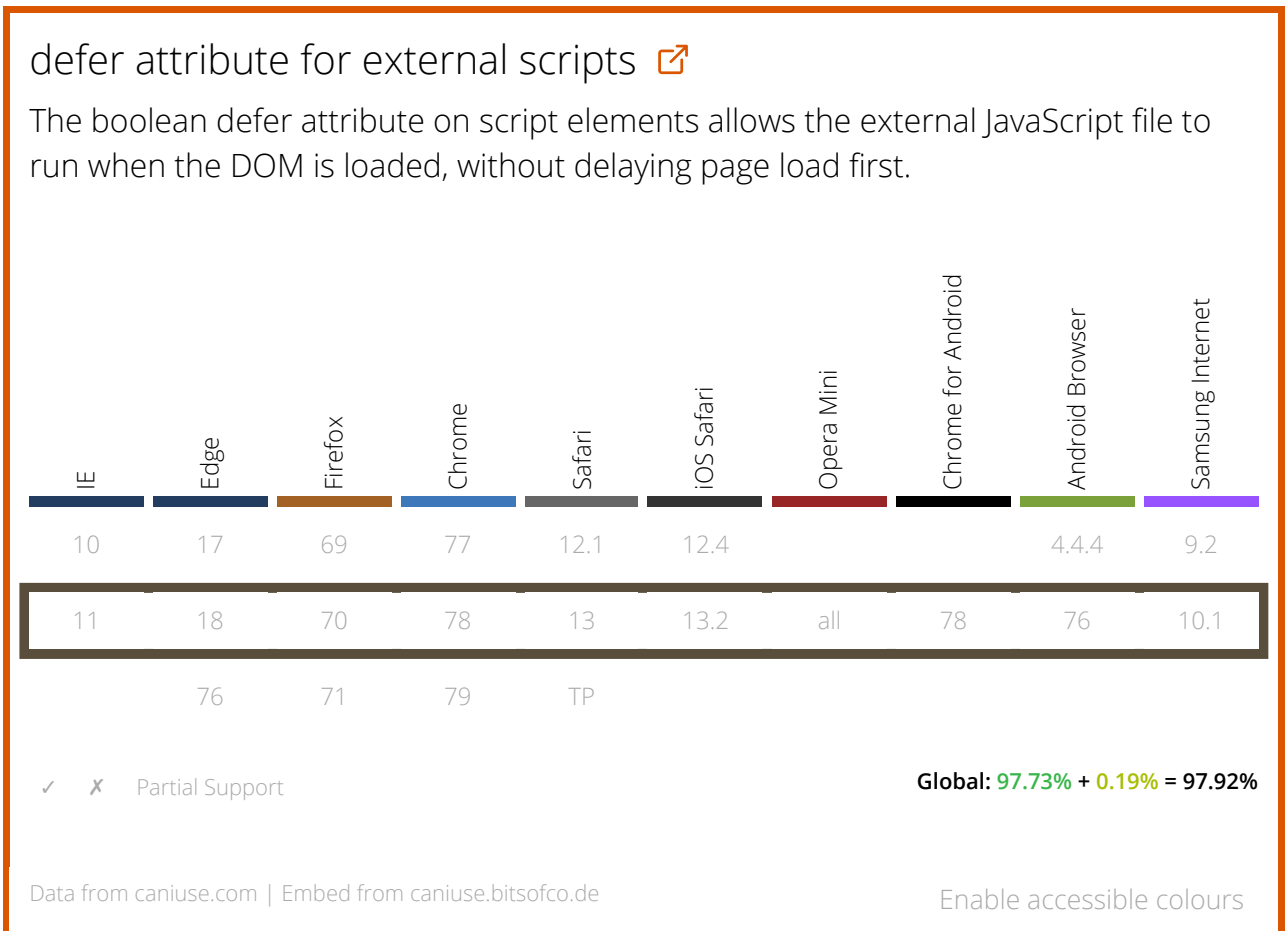
### Is the script a (small) dependency?

Finally, if the script is relatively small, and/or is depended on by other files, it may be more useful to have it defined inline. Although having it inline will block the parsing of the HTML document, it should not be a significant interference if it's a small size. Additionally, if it is depended on by other files, the minor blocking may be necessary.

## Support and Modern Browser Engines

The support for the **async** and **defer** attributes is very widespread -






It is worth noting that the behaviour of these attributes may be slightly different across different JavaScript engines. For example, in V8 (used in Chromium), an attempt is made to parse all scripts, regardless of their attributes, on a separate dedicated thread for script execution. This way, the "parser blocking" nature of JavaScript files should be minimised as a default.

---