**Async vs Defer**

ASYNC and DEFER are similar in that they allow scripts to load without blocking the HTML parser which means users see page content more quickly.

There are 2 differences:

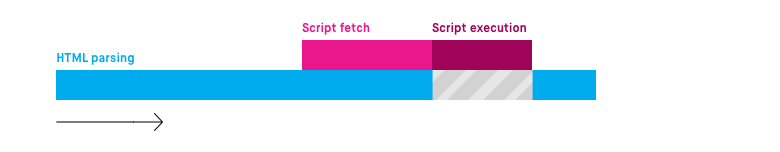
1. Scripts loaded with ASYNC are parsed and executed immediately when the resource is done downloading. Whereas DEFER scripts don’t execute until the HTML document is done being parsed
2. .ASYNC scripts may load out-of-order, whereas DEFER scripts are executed in the order in which they appear in markup.

**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.

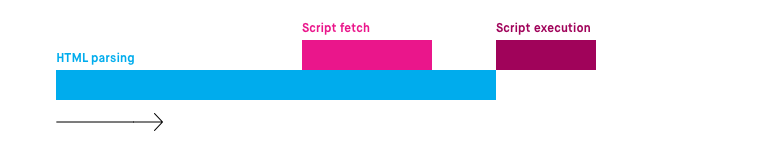


**The defer Attribute**

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.



Reference：

https://bitsofco.de/async-vs-defer/

Reference: <https://calendar.perfplanet.com/2016/prefer-defer-over-async/>

**ReadyState**

The current state of the document can be accessed using readyState property of the document object. document.readyState tells us the status of the page load.

There are 3 different possible states:

1. loading — The document is loading (the .html file is being downloaded/parsed)
2. interactive — In this state, the DOM is loaded and accessible. However, resources like images, stylesheets, and JavaScript files have not finished downloading/loading/parsing. The DOMContentLoaded event is also fired when the readyState changes from loading to interactive.
3. complete — the document and all resources like images/stylesheets have finished loading.

Reference:

<https://levelup.gitconnected.com/understand-browser-readystate-and-how-to-track-the-interactivity-of-the-content-on-your-web-page-8d2802f29aa>

Time complexity

Time complexity is the amount of time taken by an algorithm to run, as a function of the length of the input. It measures the time taken to execute each statement of code in an algorithm.

There are different types of time complexities used, let’s see one by one:

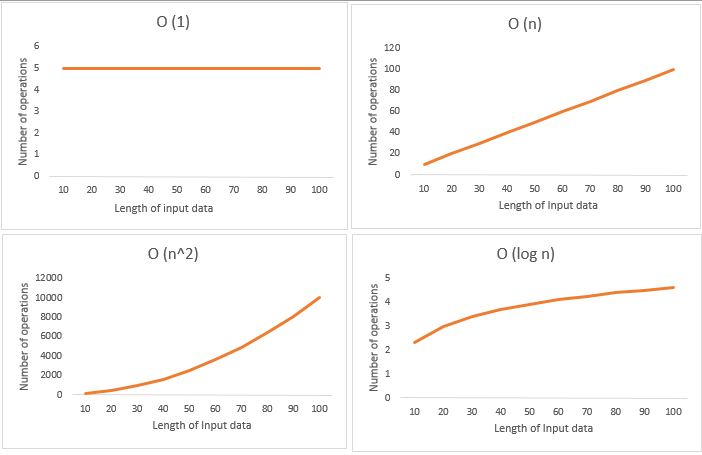
**1. Constant time – O (1)**

**2. Linear time – O (n)**

**3. Logarithmic time – O (log n)**

**4. Quadratic time – O (n^2)**

**5. Cubic time – O (n^3)**



**reference:**

https://www.mygreatlearning.com/blog/why-is-time-complexity-essential/#t3