* - [Narrator] When you open an HTML document in the browser, the browser will read that document, line by line from the top down and fetch and execute any elements, it encounters as necessary .
* In a typical header for an HTML document, we may have references to external style sheets, and two JavaScript files .
* And both of these are needed for the page to render properly .
* So as the browser encounters these, it'll stop what it's doing, go to get the external file, then execute whatever's in that file, and then continue the rendering .
* When it comes to JavaScript, this can cause some significant issues .
* For example, we've already seen this rendering problem, where the JavaScript is referenced, and run in the browser, before the elements it's acting upon have been rendered, and therefore, we get an error in the browser .
* It can't do what the JavaScript is trying to do, because the elements don't yet exist .
* The traditional solution to this problem is to simply move, the script tag to the end of the document, so the browser only encounters it, when it's rendered everything else .
* This is not a solution though, this is a hack, and it introduces a whole range of other problems, there's a good chance you have JavaScript that should run at the beginning, or should happen, while the document is being loaded .
* And doing it this way, would mean you have to just split up the JavaScript depending on what it's doing, and then place some of it at the top, and some on the bottom .
* It's a really imprecise and hacky way of doing it, and it doesn't really work all that well .
* *For this reason, we now have new tools in JavaScript to tightly control how, and when JavaScript is loaded* .
* These tools are the **async and defer keywords**, and they need a bit of an explanation .
* So this is the default behavior .
* ***If you just reference a script and point to the file, this is what happens, the browser will start parsing the HTML document, up until the point where it finds this reference, then it stops parsing the HTML, goes and downloads the JavaScript file, then executes whatever's in the JavaScript file, and then picks up the HTML parsing .***
* This is called **content or render blocking**, because it quite literally, blocks the rendering of the content in the page .

Timeline

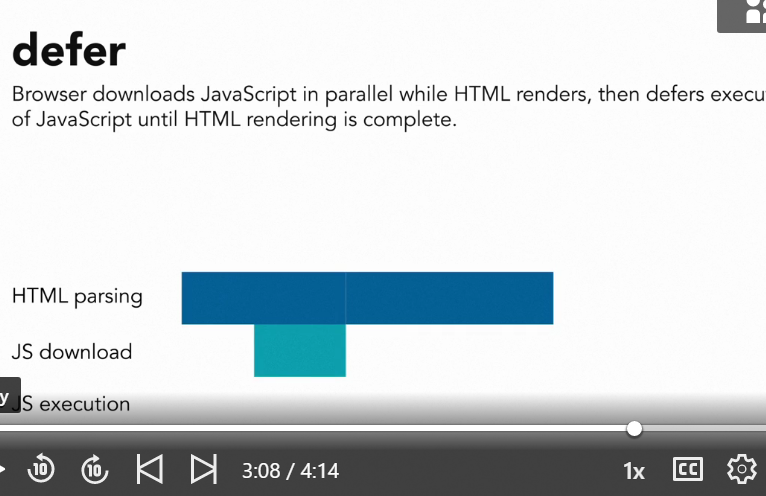
Description automatically generated with medium confidence

* And it can cause all sorts of weird problems, like what you saw in my example, and it also causes the page to just load slower, and it's not a great solution .
* **The async keyword changes this behavior significantly .**
* *It tells the browser, 'Hey, when you encounter a JavaScript file, keep parsing the HTML while you download the JavaScript file, and only stop the parsing when you actually have the file, then execute whatever's in the JavaScript file, and then continue HTML parsing .*
* *' This dramatically shortens the time it takes for the browser to execute everything .*

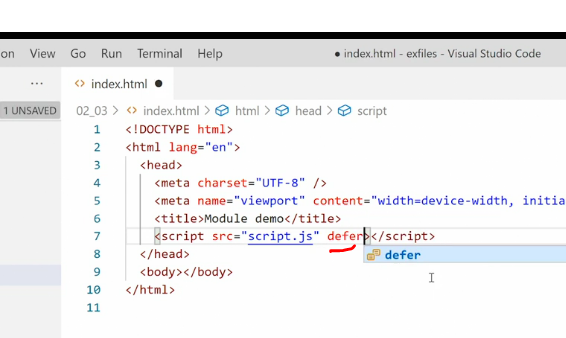
A picture containing graphical user interface

Description automatically generated

* And it doesn't, create this huge render blocking issue .
* There's still a hole in the rendering, but it's much shorter .
* This is good for some purposes, especially when you need, to get the JavaScript to parse as quickly as possible, and you don't really care about render blocking .
* But in our circumstance, we need to make sure the browser, only executes the JavaScript, after the document is complete .
* **For that we have the defer keyword** .
* The defer keyword tells the browser, 'parse your HTML and if you encounter JavaScript, just load it alongside your HTML parsing .



* *Then when the HTML parsing is complete, execute whatever JavaScript you have .*
* ' So this is exactly what we want for our purposes .
* That means, I can go back to my script, and then at the end of my script tag, just add defer, save the file, and in the browser, everything is now working properly, because we are literally deferring the execution, off the script until, everything else has been rendered .



* Here's your takeaway .
* *Async and defer should be the standard way of loading JavaScript today .*

Text

Description automatically generated

* Only use render blocking, meaning ,either placing the script tag without the async and defer keywords, or placing the, script tag at the bottom of the document, if you have a specific reason to do so .



* And trust me, I've asked around and tried to figure it out myself, I have yet to find a use case, where it makes sense, to not use async and defer .
* Loading JavaScript in the footer, is now an anti pattern .
* From here on forward, JavaScript should always be loaded in the head, and then you use async or defer, to control when that JavaScript is executed on the document .