**Asynchronous JavaScript - Part 5… -** [Marc Kirk](https://medium.com/@byteslovesbits?source=post_page-----b66598d17cc--------------------------------) Mar 3, 2022

**The Event Loop**

We now arrive at what is perhaps the most critical aspect of understanding asynchronous JavaScript — the event loop.

One way to understand the event loop is to see how Chrome’s V8 engine works. Copy the code from Figure 1 into Chrome’s developer tools console and press Enter. This will take us to a debugging environment where we can analyse V8’s order of execution. Figure 2.

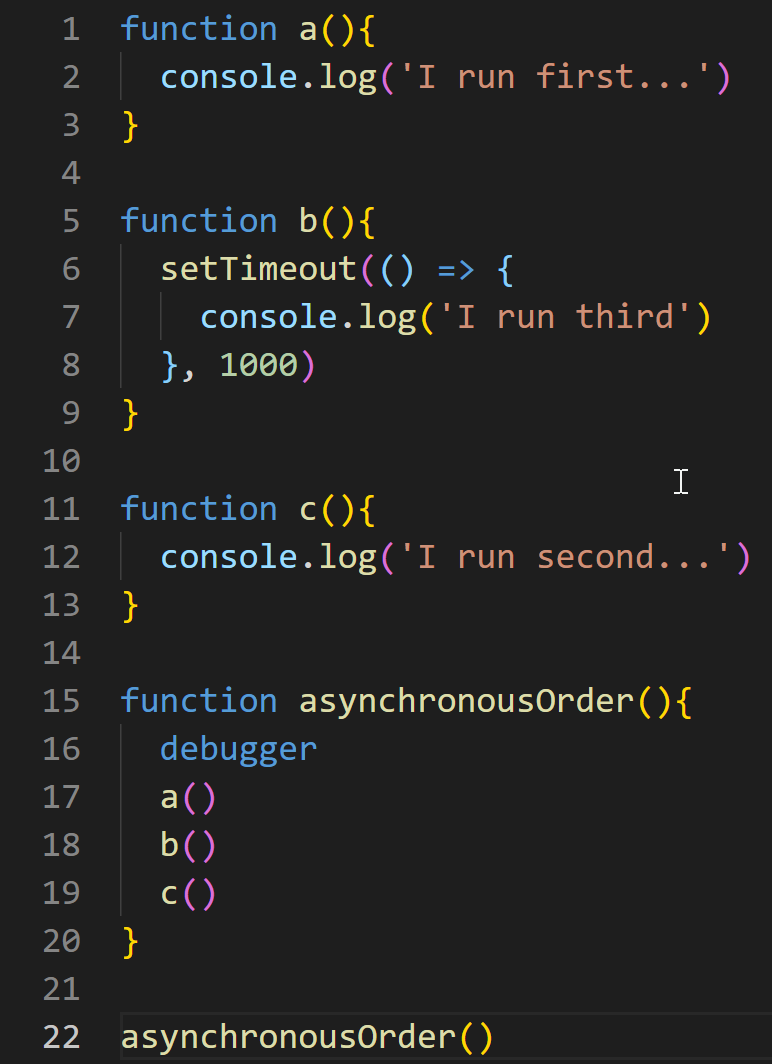


Figure 1 — asynchronous execution order

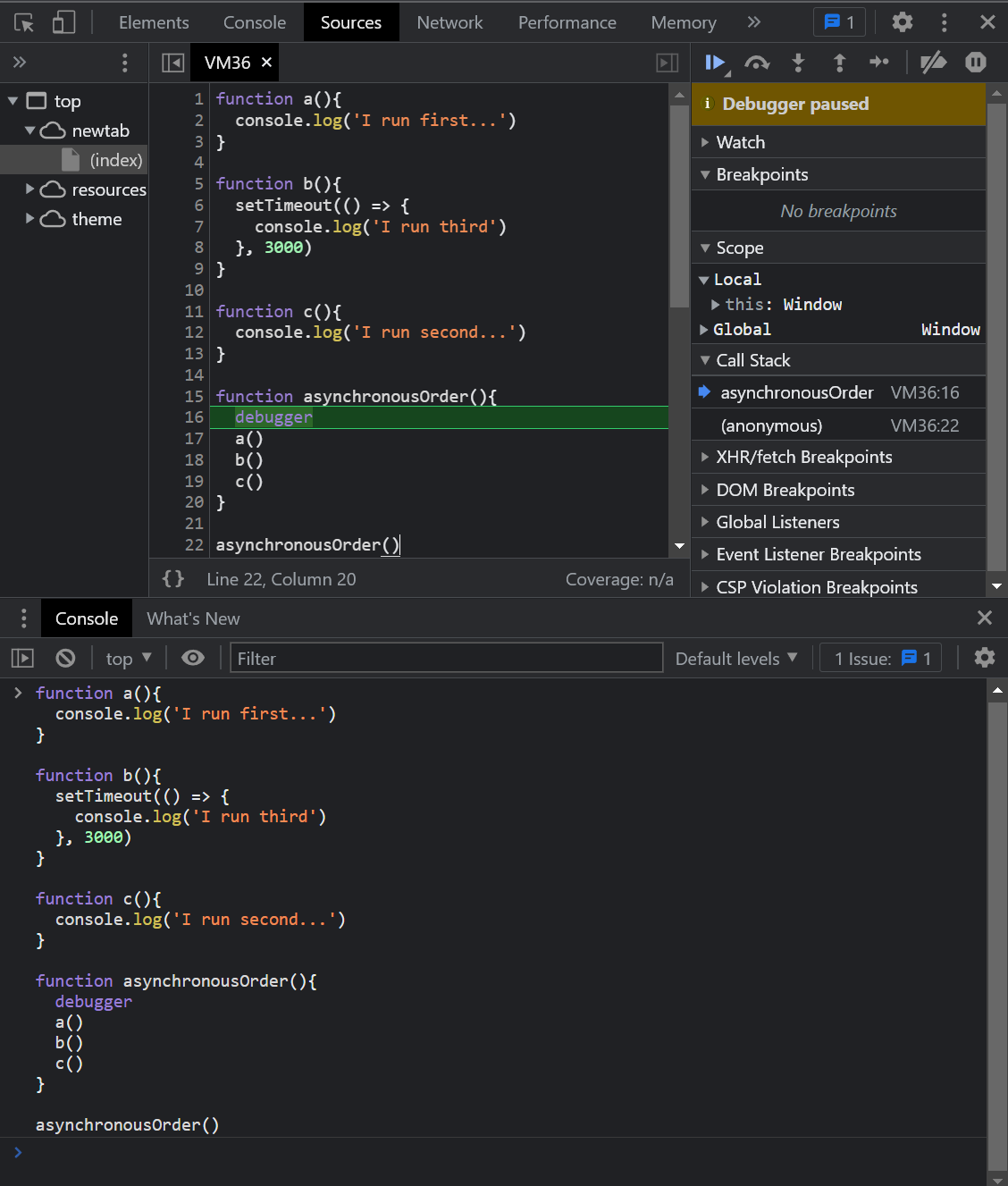


Figure 2 — enter the code into the console and press enter to open a debugging environment.

**Order of Operations**

The debugging environment from Figure 2 provides us with a high level view of V8’s behaviour.

First, the **main()** or **anonymous()**function is pushed onto the stack.

V8 will then execute code synchronously until it reaches line 6.

On line 6, the browser turns synchronous JavaScript into asynchronous JavaScript with the help of the global web-api function named **setimeout.**

The browser will then set a timer for 1000 ms and once that 1000 ms has elapsed, **it will execute the code within the function that is passed to to setimeout**. In this case, the function passed into **setimeout** is an anonymous arrow function. In the meantime, V8 will proceed synchronously. It will execute **c()**as the **setimeout**function counts down from 1000 ms. This process is best illustrated using a GIF. Figure 3. Pay attention to the Call Stack area of the debugging window.

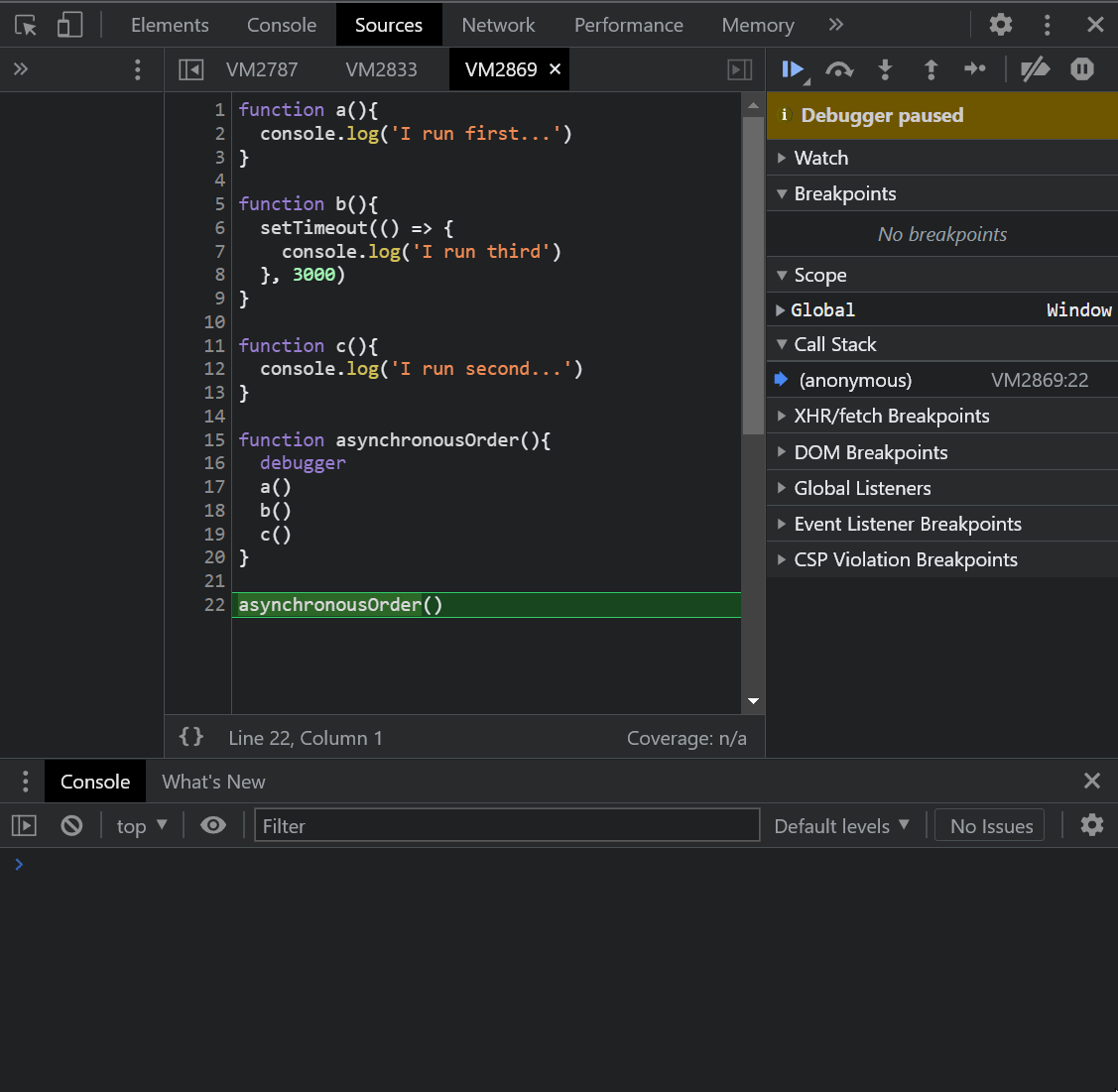


Figure 3 — analysing V8’s asynchronous behaviour

To further illustrate this concept, take a look at the GIF in Figure 4. The GIF asynchronously updates the DOM using a similar method to **setimeout** — named **setinterval.**Think of **setinterval**as calling **setimeout**over and over — **Ad infinitum.**Every second, the DOM is asynchronously updated to change the bodies background color to a random colour from within the colors array.

Also, the page is non-blocking, meaning we can interact with it! **In essence, the browser gives the illusion that JavaScript is no longer a blocking, single threaded language.**As **setinterval** counts down, the page is free to do other asynchronous stuff, such as fetching data from a server.

The browser gives the illusion that synchronous, blocking, single threaded JavaScript is asynchronous JavaScript through the use of web-apis and the event model.

***setinterval****and****setimeout****are asynchronous functions that are part of web-apis that are implemented by the browser vendor.*

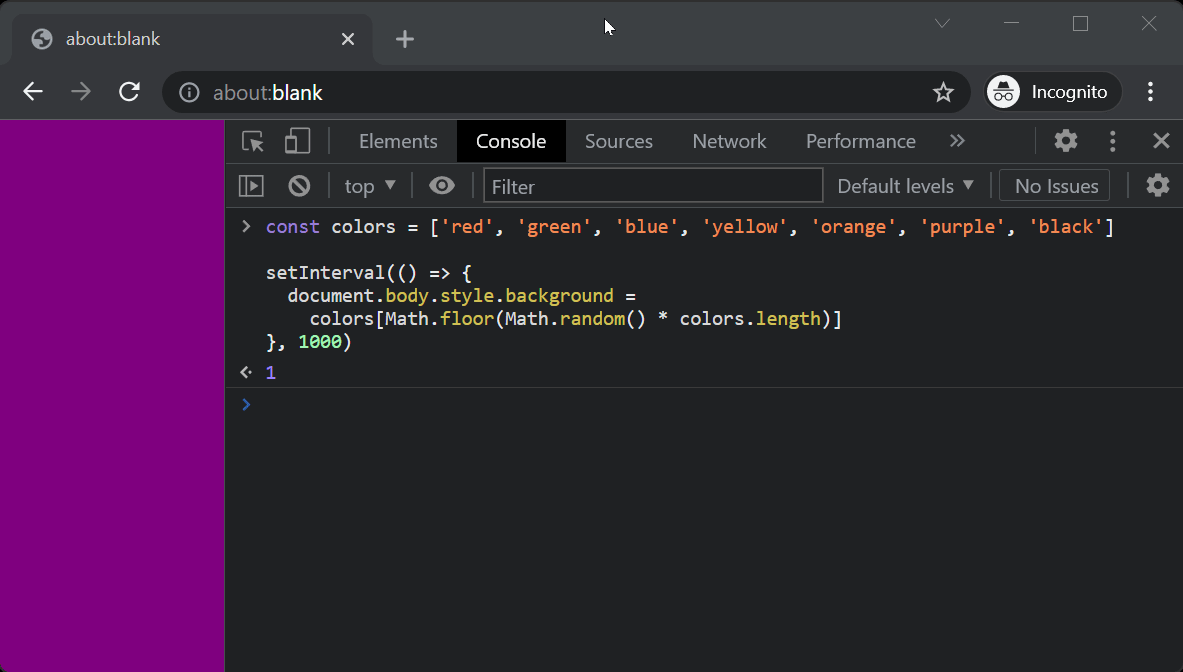


Figure 4 — asynchronously updating the DOM every second

An interesting point about **setimeout**and **setinterval** is that these functions are not implemented by V8. If you were to grab the source code for V8 and search for **setimeout**, you would not find it. It is the responsibly of the various browser vendors such as Chrome and Firefox to implement these functions according to the HTML standard. Details of the event loop can are found in section [8.1.6](https://html.spec.whatwg.org/multipage/webappapis.html#event-loops) of the HTML standard. Here you will also find detailed information on web-apis.

*For details how browser vendors implement the event loop, head over to section*[*8.1.6*](https://html.spec.whatwg.org/multipage/webappapis.html#event-loops)*of the HTML standard.*

*For details how browser vendors implement web-apis, head over the section*[*8*](https://html.spec.whatwg.org/multipage/webappapis.html#webappapis)*of the HTML standard.*