Brian Barbour
Posted on 3 Jun 2019

If Javascript Is Single Threaded, How Is It Asynchronous?

#javascript #webdev #beginners

Javascript is a single threaded language. This means it has one call stack and one memory heap. As expected, it executes code in order and must finish executing a piece code before moving onto the next. It's synchronous, but at times that can be harmful. For example, if a function takes awhile to execute or has to wait on something, it freezes everything up in the meanwhile.

A good example of this happening is the window alert function. `alert("Hello World")`

You can't interact with the webpage at all until you hit OK and dismiss the alert. You're stuck.

So how do we get asynchronous code with Javascript then?

Well, we can thank the Javascript engine (V8, Spidermonkey, JavaScriptCore, etc...) for that, which has Web API that handle these tasks in the background. The call stack recognizes functions of the Web API and hands them off to be handled by the browser. Once those tasks are finished by the browser, they return and are pushed onto the stack as a callback.

Open your console and type `window` then press enter. You'll see most everything the Web API has to offer. This includes things like ajax calls, event listeners, the fetch API, and setTimeout. Javascript uses low level programming languages like C++ to perform these behind the scenes.

Let's look at a simple example, run this code your console:

```
console.log("first")
setTimeout(() => {
    console.log("second")
}, 1000)
```

Next, the Javascript engine's event loop kicks in, like a little kid asking "Are we there yet?" on a road trip. It starts firing, waiting for events to be pushed into it. Since the `setTimeout` isn't finished, it returns `undefined`, as the default, well because it hasn't been given the value yet. Once the callback finally does hits we get `console.log("second")` printed.

There's a really good site that slows this all down and shows this happening.

http://latentflip.com/loupe

I suggest playing around in this sandbox to help solidify your understanding. It helped me get a feel for how asynchronous code can work with Javascript being single threaded.

👋 Kindness is contagious

Discover a treasure trove of wisdom within this insightful piece, highly respected in the nurturing DEV Community enviroment. Developers, whether novice or expert, are encouraged to participate and add to our shared knowledge basin.

A simple "thank you" can illuminate someone's day. Express your appreciation in the comments section!

On DEV, sharing ideas smoothens our journey and strengthens our community ties. Learn something useful? Offering a quick thanks to the author is deeply appreciated.

Okay

```
  setTimeout(() => console.log(item), item);
})
// 10 20 35 100 500
```

🤣🤣🤣

Bradley Griffith • 6 Jun 19 • Edited

did a flex-box version: jsfiddle.net/bradleygriffith/2dsag...

```
<div class="sorted-list" id="my-list"></div>
```

```
.sorted-list {
  align-items: flex-start;
  display: flex;
  margin: 0 -5px;
}

.sorted-list-item {
  margin: 0 5px;
}
```

```
const listEl = document.getElementById("my-list");
const arr = [10, 100, 500, 20, 35];

arr.forEach(n => {
  const itemEl = document.createElement("div");

  itemEl.className = "sorted-list-item";
  itemEl.innerHTML = n;
  itemEl.style.order = n;

  listEl.appendChild(itemEl);
});
```

Unfortunately, this cute sorting algorithm doesn't work on values that < 1.

Eugene Karataev • 4 Jun 19

It also poorly works with floats (ex. `[1.5, 1.4, 1.3, 1.2, 1.1]`) and big numbers 😄

SAURABH BAKOLIA • 26 Jul

Try with this example: [10, -100, 500, -20, 35];

Brian Barbour • 4 Jun 19

In the course I'm doing we had to use setTimeout as a way to avoid stack overflow. I don't think I'd ever do it in a real app, but it was an interesting trick.

timepp • 17 Aug 23

This is mostly equivlant with the following code

```
let arr = [10, 100, 500, 20, 35];
let sorted = 0
for (let i = 0; sorted < arr.length; i++) {
    for (const v of arr) {
        if (v === i) {
            console.log(v);
            sorted++;
        }
    }
}
```

Nandan Kumar • 4 Jun 19

Hey Eugene,
Would you please help me understand how does this happen. Any link or reference would be very

Yes, indeed it's a joke and should not be used anywhere in production. But it's nice to know how things work, It took me some time to figure out but it was worth it.
Thanks 😁

Nando • 8 Jun 19

I think you are a bit crazy, bro ;)

Abhishek Bhardwaj • 17 Nov 20

It has limitation that if any item of array contained number in billion or millions it will keep in waiting unless the time finish.

Syed Aqeel • 7 Jul 21

This is called timeout sort I think :D

James Q Quick • 17 Jun 21

This is amazing lol

Ryan Mattos • 12 Aug 22

genius

Prabhjeet Singh • 15 Jul 20

how does it sort with timeout? should it not pick items with index

Prahlad Yeri • 3 Jun 19 • Edited

Not all apps are benefited by javascript's async nature though, only those which are I/O centric. Apps which are more CPU centric like those involving statistical computations or heavy algorithms

Exactly, multi-threading or parallel computing is the key when it comes to a lot of tasks. In fact, to take the full advantage of the 4 cores of your CPU, multi-threading is a must. Async will never be able to do that however efficient it may otherwise be.

It all comes down to what your app needs to do. I/O bound operations are where async shines and you should make full use of that if your app is majorly I/O bound.

Sung M. Kim • 4 Jun 19

I've tried with setTimeout delay of 0, which I expected to run before `third`.

```
console.log("first")
setTimeout(() => {
    console.log("second")
}, 0)
console.log("third")
first
third
undefined
second
```

but I was surprised to see that `second` was returned last.

Oscar • 5 Jun 19 • Edited

The function in your timeout gets queued as a task. The script runs and once it is done (`console.log('third')`), the engine can handle the task queue and will execute the timeout callback. So, even though the timeout is zero, the function will not get called immediately.

There is a lot more to the topic and Jake Archibald wrote an amazing article about how this works. I highly recommend reading it:

jakearchibald.com/2015/tasks-micro...

Sung M. Kim • 6 Jun 19

Yogi Wisesa • 18 Aug 20

Hey Brian, coming from the future here. I'm a little bit confused by this statement "Well, we can thank the Javascript engine (V8, Spidermonkey, JavaScriptCore, etc...) for that, which has Web API that handle these tasks in the background. ", if the task is handled in the background so it's mean if the javascript isn't single thread right? since the code is executed in the same time. thank you!

Mostafa • 14 Apr

Mastering this topic I considered as a fundamental of being a JavaScript developer, knowing more about JavaScript engine (compiler) , the browser mechanisms and critical rendering path (CRP) is very important thing! I advise the developers who want to be a rock in web development field, please go through this topics and learn them carefully and on top of that make a deep understanding of network layer like http requests, responses, cache, cookies, storage, sessions. And related stuff to a browser network.

Aleksandar15 • 11 Jun 22

I think you are wrong at this part `...we can thank the Javascript engine (V8, Spidermonkey, JavaScriptCore, etc...) for that, which has Web API....`

We don't thank JS Engine for that, because Web API is not part of the JS Engine but the `JS runtime environment provided by the Browser`, also provided by the browser is the JS Engine itself (V8 for Chrome) and Callback Queue and the Event Loop.
The JS (V8) Engine is made up of Memory Heap & Call Stack.

Furthermore, the purpose of the JS engine is to parse/translate source code that we developers write into machine code that allows a computer to perform specific tasks on its hardware level.

Britain Green • 3 Jun 19

Great, Brian!

Nice article Brian with a practical example.

Here is another resource which can be a good primer for newbies in understanding why Javascript is called single-threaded language w3spot.com/2020/07/how-asynchronou....

View full discussion (45 comments)

Code of Conduct • Report abuse

👋 Kindness is contagious

Discover a treasure trove of wisdom within this insightful piece, highly respected in the nurturing DEV Community enviroment. Developers, whether novice or expert, are encouraged to participate and add to our shared knowledge basin.

A simple "thank you" can illuminate someone's day. Express your appreciation in the comments section!

On DEV, sharing ideas smoothens our journey and strengthens our community ties. Learn something useful? Offering a quick thanks to the author is deeply appreciated.

Okay

LOCATION
Rochester, NY

EDUCATION
14+ Years In Tech Industry

PRONOUNS
He

WORK
Software Engineer and Writer

JOINED
9 Apr 2019

More from Brian Barbour

Believe In The Web
#webdev #javascript #html #software

When to Branch Your Code (Git)
#git #softwareengineering #beginners #coding

Creating a MERN stack app that uses Firebase Authentication - Part Two
#javascript #webdev #node #react