Is setTimeout a good solution to do async functions with javascript?

Asked 5 years, 9 months ago Active 8 days ago Viewed 49k times

50

14

Searching in the web about async functions, I found many articles using setTimeout to do this work:

```
window.setTimeout(function() {
   console.log("second");
}, 0);
console.log("first");
```

Output:

```
first
second
```

This works, but is a best practice?

javascript jquery

edited Apr 2 '18 at 18:51
Uwe Keim
28.2k 32 141 226

asked Oct 28 '13 at 3:16
viniciuswebdev
575 1 5 15

2 It's certainly the simplest technique. However, if you're actually trying to run the asynchronous task in a separate thread, web workers are more effective. (HTML 5) – Flight Odyssey Oct 28 '13 at 3:19

2 @minitech: That's exactly my point (that `setTimeout` runs in a single thread, while web workers do not). They serve different purposes, and often people use `setTimeout` when web workers are really what they're looking for. Without knowing more details about what the OP is trying to accomplish, I'm not sure which is best for this case. – Flight Odyssey Oct 28 '13 at 3:24

1 Practice? Nevermind that. What's your goal? Tell us that and then we can help you best achieve it. – Joe Simmons Oct 28 '13 at 3:24

@FlightOdyssey: Oh. I didn't quite get the meaning of "more effective" there. :P – Ry- ♦ Oct 28 '13 at 3:26

1 Async wont have better performance. You're just delaying what's going to happen. If your code needs to wait before writing, then sure, keep it. – Joe Simmons Oct 28 '13 at 13:31

3 Answers

64

`setTimeout(function(){...}, 0)` simply queues the code to run once the current call stack is finished executing. This can be useful for some things.

So yes, it's asynchronous in that it breaks the synchronous flow, but it's not actually going to execute concurrently/on a separate thread. If your goal is background processing, have a look at webworkers. There's also a way to use iframes for background processing.

Update:

To further clarify, there's a difference between concurrency/backgrounding and asynchronous-ness. When code is asynchronous that simply means it isn't executed sequentially. Consider:

```
var foo='poo';
setTimeout(function() {
  foo='bar'
}, 100);
console.log(foo);
```

Run code snippet   Hide results   Full page

```
poo                                                    11:26:44.559
```

The value 'poo' will be alerted because the code was not executed sequentially. The 'bar' value was assigned asynchronously. If you need to alert the value of `foo` when that asynchronous assignment happens, use callbacks:

```
/* contrived example alert */
var foo = 'poo';

function setFoo(callback) {
  setTimeout(function() {
    foo = 'bar';
    callback();
  }, 100);
};
setFoo(function() {
  console.log(foo);
});
```

Run code snippet   Hide results   Full page

```
bar                                                                11:27:17.241
```

So yes, there's some asynchronous-ness happening above, but it's all happening in one thread so there are no performance benefits.

When an operation takes a long time, it is best to do it in the background. In most languages this is done by executing the operation on a new thread or process. In (browser) javascript, we don't have the ability to create new threads, but can use webworkers or iframes. Since this code running in the background breaks the sequential flow of things it is asynchronous.

TLDR: All backgrounded/concurrent code happens asynchronously, but not all asynchronous code is happening concurrently.

See Also: Understanding Asynchronous Code in Layman's terms

edited Jul 23 at 12:56
H. Pauwelyn
6,226 21 51 95

answered Oct 28 '13 at 3:35
tybro0103
27.5k 29 131 161

1

```
var foo = 'poo';
setTimeout(function() {foo = 'bar'}, 100);
alert(foo);
```

A small correction to @tybro0103 's answer, during the execution of 'alert(foo)' the value 'poo' will not change because the code was not executed sequentially. The 'bar' value was assigned asynchronously and it will execute only after 100 millisecond, by that time alert will be executed.

The value of **foo** remains unaltered, **during the execution of line alert(foo)**. And will change later on time. Check @vishal-lia comment.

edited Apr 4 '18 at 5:39

answered Nov 26 '15 at 9:09
Vaisakh Rajagopal
397 3 15

1 * you meant the value 'bar' will not be alerted because `alert` already triggered `foo` – Paul Jan 10 '16 at 17:08

Yes correct, you can easily verify it by executing the same on console. – Vaisakh Rajagopal Mar 3 '17 at 10:53

I changed the number value in the setTimeout(..) to 0. It turns out it still prints 'poo' i.e. the value has not changed. Could you please explain why? – Sameer Khanal Dec 18 '17 at 6:34

1 @VaisakhRajagopal ok got it so the setTimeout function runs once the current stack is complete plus the number of milliseconds specified (as opposed to just the milliseconds specified). Thanks! – Sameer Khanal Dec 20 '17 at 0:26

1 When you say "The value of foo remains unaltered." you are actually confusing people. foo definitely changes to 'bar' but after 100 ms, if you log 'foo' it will still print 'poo' because "function() {foo = 'bar'}" is called after 100ms. You can verify in browser console to check change in value. – Vishal-Lia Apr 3 '18 at 15:56

0

By default, JavaScript is asynchronous whenever it encounters an async function, it queued that function for later. But if you want a pause js for you can do it use promises Case 1: output hello(will not wait for setTimeout) https://jsfiddle.net/shashankgpt270/h0vr53qy/

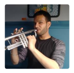
```
//async
function myFunction() {
let result1='hello'
//promise =new Promise((resolve,reject)=>{
setTimeout(function(){
```

```
resolve("done");
result1="done1";
}, 3000);
//});
 //result = await promise
 alert(result1);
}
myFunction();
```

case 2: output done1(will wait for setTimeout) https://jsfiddle.net/shashankgpt270/1o79fudt/

```
async function myFunction() {
let result1='hello'
promise =new Promise((resolve,reject)=>{
setTimeout(function(){
resolve("done");
result1="done1";
}, 3000);
});
 result = await promise
 alert(result1);
}
myFunction();
```

answered Jan 18 at 15:55


Shashank Gupta
1