Is JavaScript multithreaded?

Asked 15 years, 1 month ago Modified 9 years, 2 months ago Viewed 55k times



Here's my issue - I need to dynamically download several scripts using jQuery.getScript() and execute certain <u>JavaScript</u> code after all the scripts were loaded, so my plan was to do something like this:



64





```
function GetScripts(scripts, callback)
{
  var len = scripts.length
  for (var i in scripts)
  {
    jQuery.getScript(scripts[i], function()
    {
        len --;
        // executing callback function if this is the last script that loaded
        if (len == 0)
            callback()
        })
    }
}
```

This will only work reliably if we assume that script.onload events for each script fire and execute sequentially and synchronously, so there would never be a situation when two or more of the event handlers would pass check for (len == 0) and execute callback method.

So my question - is that assumption correct and if not, what's the way to achieve what I am trying to do?

javascript jquery

Share

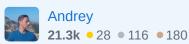
Improve this question

Follow

edited Nov 2, 2009 at 20:02



asked Nov 2, 2009 at 19:19



- javascript is multithreaded, heres a link to check out to show an example: <u>ajaxian.com/archives/multi-threaded-javascript</u> – Anthony Forloney Nov 2, 2009 at 19:23
- @aforloney that's not exactly right. As the Sun engineer cited in the article says, the Rhino JavaScript engine "allows Java method calls. So, we can use that to create script wrappers for Java platform classes... we can easily create threads in script." Jeff Sternal Nov 2, 2009 at 20:22

9 Answers

Sorted by:

Highest score (default)

\$



88

No, JavaScript is not multi-threaded. It is event driven and your assumption of the events firing sequentially (assuming they load sequentially) is what you will see. Your current implementation appears correct. I believe jQuery's <code>.getScript()</code> injects a new <code><script></code> tag, which should also force them to load in the correct order.



Share Improve this answer Follow

answered Nov 2, 2009 at 19:22







4

I basically don't care whether about the order of loading, I just need to make sure that my callback function will only get executed when all of the scripts are loaded, because it depends on all of them. – Andrey Nov 2, 2009 at 19:27

- 9 This is probably an embarrassing simple-minded question, but if JavaScript is not multithreaded, why do multiple chained jQuery animations (for example) appear to happen simultaneously instead of in sequence? – Larry Lustig Nov 2, 2009 at 19:32
- 12 The animation updates are triggered by timer events being fired. Jacob Nov 2, 2009 at 19:37
- As far as I understand, animations are using timers, and each animation is not a single process but a sequence of small incremental changes on some timer event, hence it seems like multiple animations happen simultaneously Andrey Nov 2, 2009 at 19:37



36

Currently JavaScript is not multithreaded, but the things will change in near future. There is a <u>new thing in HTML5</u> called <u>Worker</u>. It allows you to do some job in background.



But it's currently is not supported by all browsers.



Share

edited Nov 2, 2009 at 21:06

answered Nov 2, 2009 at 19:33



Improve this answer



Peter Mortensen **31.6k** • 22 • 109 • 133



Ivan Nevostruev **28.7k** • 8 • 67 • 82

Follow

Worker is a new thing in Javascript implementation, but I don't see how it has anything to do with HTML 5 -Andrey Nov 2, 2009 at 19:39

- Web Workers is part of the HTML5 specification. <u>whatwg.org/specs/web-workers/current-work</u>
 Rob Nov 2, 2009 at 19:52
- 8 Two things to note about Workers is that they do not have access to the DOM and cannot access the parent page. Seanny123 Sep 10, 2013 at 6:30
- Now workers are supported in all new browsers (IE 10 was the last to get on board) w3schools.com/html/html5 webworkers.asp CodeMonkey Nov 21, 2016 at 17:47



The <u>JavaScript</u> (<u>ECMAScript</u>) <u>specification</u> does not define any threading or synchronization mechanisms.

18



Moreover, the JavaScript engines in our browsers are deliberately single-threaded, in part because allowing more than one UI thread to operate concurrently would open an enormous can of worms. So your assumption and implementation are correct.

As a sidenote, <u>another commenter</u> alluded to the fact that any JavaScriptengine vendor could add threading and synchronization features, or a vendor could enable users to implement those features themselves, as described in this article: <u>Multi-threaded JavaScript</u>?

Share

Improve this answer

Follow

edited May 23, 2017 at 11:46



answered Nov 2, 2009 at 20:44



1 I'm always impressed with an answer that goes to the spec. ps. (sincerely, not sarcasm) – Michael Easter Nov 2, 2009 at 21:11



14

JavaScript is absolutely not multithreaded - you have a guarantee that any handler you use will not be interrupted by another event. Any other events, like mouse clicks, XMLHttpRequest returns, and timers will queue up while your code is executing, and run one after another.



Share

Improve this answer

Follow

edited Nov 2, 2009 at 21:05



answered Nov 2, 2009 at 20:38





No, all the browsers give you only one thread for JavaScript.



Share

Improve this answer







Follow



+1, though they could do otherwise, but for the havoc that would ensue. – Jeff Sternal Nov 2, 2009 at 20:24



To be clear, the **browser JS implementation** is **not multithreaded**.



The language, JS, can be multi-threaded.



The question does not apply here however.

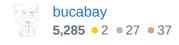


What applies is that getScript() is asynchronous (returns immediately and get's queued), however, the browser will execute DOM attached <script> content sequentially so your dependent JS code will see them loaded sequentially. This is a browser feature and not dependent on the JS threading or the getScript() call.

If getScript() retrieved scripts with xmlHTTPRequest, setTimeout(), websockets or any other async call then your scripts would not be guaranteed to execute in order. However, your callback would still get called after all scripts execute since the execution context of your 'len' variable is in a closure which persists it's context through asynchronous invocations of your function.

Share Improve this answer Follow

answered Nov 11, 2014 at 9:09





JS in general is single threaded. However HTML5 Web workers introduce multithreading. Read more at http://www.html5rocks.com/en/tutorials/workers/basics/



Share Improve this answer Follow













Thought it might be interesting to try this out with a "forced", delayed script delivery ...

1







2. added delayis.php as the 2nd array element. delayis.php sleeps for 5 seconds before delivering an empty is object.

1. added two available scripts from google

- 3. added a callback that "verifies" the existence of the expected objects from the script files.
- 4. added a few js commands that are executed on the line after the GetScripts() call, to "test" sequential is commands.

The result with the script load is as expected; the callback is triggered only after the last script has loaded. What surprised me was that the js commands that followed the GetScripts() call triggered without the need to wait for the last script to load. I was under the impression that no js commands would be executed while the browser was waiting on a js script to load ...

```
var scripts = [];
scripts.push('http://ajax.googleapis.com/ajax/libs/prototype/1.6.1.0/prototype.j
scripts.push('http://localhost/delayjs.php');
scripts.push('http://ajax.googleapis.com/ajax/libs/scriptaculous/1.8.3/scriptacu
function logem() {
    console.log(typeof Prototype);
    console.log(typeof Scriptaculous);
    console.log(typeof delayedjs);
}
GetScripts( scripts, logem );
console.log('Try to do something before GetScripts finishes.\n');
$('#testdiv').text('test content');
<?php
sleep(5);
echo 'var delayedjs = {};';
```

Share Improve this answer Follow



Well, don't confuse yourself - browser downloads scripts outside of javascript thread, hence javascript proceeds without waiting for scripts to load – Andrey Nov 2, 2009 at 22:09



0

You can probably get some kind of multithreadedness if you create a number of frames in an HTML document, and run a script in each of them, each calling a function in the main frame that should make sense of the results of those functions.



Share Improve this answer Follow

answered Sep 2, 2011 at 8:40



