What is the cleanest way to get the progress of JQuery ajax request?

In plain javascript is very simple: need just to attach the callback to `{XMLHTTPRequest}.onprogress`

```
var xhr = new XMLHttpRequest();

xhr.onprogress = function(e){
    if (e.lengthComputable)
        var percent = (e.loaded / e.total) * 100;
};

xhr.open('GET', 'http://www...', true);
xhr.onreadystatechange = function() {
    ...
};
xhr.send(null);
```

but I'm doing an ajax site that download html data with JQuery (`$.get()` or `$.ajax()`) and I was wondering which is the best way to get the progress of a request in order to display it with a little progress bar but curiously, I'm not finding anything usefull in JQuery documentation...

javascript ajax jquery xmlhttprequest

asked Oct 1 '13 at 22:21

guari
2,320 ● 2 ● 22 ● 21

3 This one looks promising dave-bond.com/blog/2010/01/JQuery-ajax-progress-HMTL5 for html5 – PSL Oct 1 '13 at 22:27

1 Ooh thanks guys! so need to override xhr.. the strange thing is that I've inspected with Chrome Dev Tools the so called `jqXHR` object (the wrapper of xhr object returned by `$.ajax()`) and found a `progress` attribute in it (along with `abort` , `complete` , `success` , etc.), but in JQuery docs this is missing: api.jquery.com/jQuery.ajax/#jqXHR – guari Oct 2 '13 at 9:14

2 github.com/englercj/jquery-ajax-progress I use this and its quite the same as other answers but I prefere to have more generic stuff – KeizerBridge Nov 5 '14 at 14:59

5 Answers

Something like this for `$.ajax` (HTML5 only though):

```
$.ajax({
    xhr: function() {
        var xhr = new window.XMLHttpRequest();
        xhr.upload.addEventListener("progress", function(evt) {
            if (evt.lengthComputable) {
                var percentComplete = evt.loaded / evt.total;
                //Do something with upload progress here
            }
       }, false);

       xhr.addEventListener("progress", function(evt) {
            if (evt.lengthComputable) {
                var percentComplete = evt.loaded / evt.total;
                //Do something with download progress
            }
       }, false);

       return xhr;
    },
    type: 'POST',
    url: "/",
    data: {},
    success: function(data){
        //Do something on success
    }
});
```

answered Oct 1 '13 at 22:27

mattytommo
45.2k ● 14 ● 92 ● 125

Looks promising, but how can this possibly work? The entire pipeline consists of three steps - sending a request, processing the request in the backend to generate some data, and return it back. How can the client side possibly know what is being done in the backend and how much time it will take that it can calculate the progress? – SexyBeast Apr 30 '17 at 21:11

The HTTP response header tells us how many bytes to expect, this progress is simply counting how many bytes have been received so far. It will stay at zero until the HTTP response is actually sent – J. Allen Aug 10 '17 at 21:10

jQuery has already implemented promises, so it's better to use this technology and not move events

```
$.ajax(url)
  .progress(function(){
    /* do some actions */
  })
  .progressUpload(function(){
    /* do something on uploading */
  });
```

Check it out at github

edited Jul 16 '16 at 12:02
shilch
75 • 11

answered Aug 28 '15 at 13:50
likerRr
515 • 4 • 12

I liked the way you use the IFI factory. I did not knew that technique! – CodeArtist Nov 25 '15 at 20:43

This is currently the best solution suggested here. – atomless Nov 27 '15 at 11:00

1 Working and elegant solution but you may be aware that it can break your existing code because it breaks all calls to the deprecated .success and .error. It also strips all non standard attributes you set on a jqXHR object. It does not provide also the context into "this" for the uploadProgress callback (maybe the same for progress but not tested) as it is done for all the standard promises for jqXHR. So you will need to pass the context in a closure. – frank Mar 13 '16 at 22:07

1 I get error: `TypeError: $.ajax(...).progress(...).progressUpload is not a function` .... What's the issue? – Universal Grasp Jun 26 '16 at 13:04

@UniversalGrasp hi, please, open an issue at github and provide information about what you've done. This library wasn't updated for ages :) may be something has changed in jQuery itself – likerRr Jun 27 '16 at 7:00

jQuery has an `AjaxSetup()` function that allows you to register global ajax handlers such as `beforeSend` and `complete` for all ajax calls as well as allow you to access the `xhr` object to do the progress that you are looking for

answered Oct 1 '13 at 22:26
Kevin Pei
4,224 • 6 • 26 • 45

2 Thanks for the link. Can you include an example in your answer? – Michael Scheper May 11 '15 at 23:51

$.ajaxSetup({ xhr: function () { console.log('setup XHR...'); } }); – Flo-Schield-Bobby May 17 '15 at 9:17

4 And an example that answers the question? I'm afraid I can't upvote an answer that leaves me doing a lot of fiddling and reading, especially when the linked page says nothing about progress. Honestly, I'm sceptical about this, especially given the warning on that page, that says 'Note: Global callback functions should be set with their respective global Ajax event handler methods— `.ajaxStart()`, `.ajaxStop()`, `.ajaxComplete()`, `.ajaxError()`, `.ajaxSuccess()`, `.ajaxSend()` —rather than within the options object for `$.ajaxSetup()`.' <api.jquery.com/jQuery.ajaxSetup/#entry-longdesc>; – Michael Scheper Aug 18 '15 at 16:01

I tried about three different ways of intercepting the construction of the Ajax object:

1. My first attempt used `xhrFields`, but that only allows for one listener, only attaches to download (not upload) progress, and requires what seems like unnecessary copy-and-paste.
2. My second attempt attached a `progress` function to the returned promise, but I had to maintain my own array of handlers. I could not find a good object to attach the handlers because one place I'd access to the XHR and another I'd have access to the jQuery XHR, but I never had access to the deferred object (only its promise).
3. My third attempt gave me direct access to the XHR for attaching handlers, but again required to much copy-and-paste code.
4. I wrapped up my third attempt and replaced jQuery's `ajax` with my own. The only potential shortcoming is you can no longer use your own `xhr()` setting. You can allow for that by checking to see if `options.xhr` is a function.

I actually call my `promise.progress` function `xhrProgress` so I can easily find it later. You might want to name it something else to separate your upload and download listeners. I hope this helps someone even if the original poster already got what he needed.

```
(function extend_jQuery_ajax_with_progress( window, jQuery, undefined ) {
    var $originalAjax = jQuery.ajax;

    jQuery.ajax = function (url, options) {
        if (typeof(url) === 'object') {
            options = url;
            url = undefined;
        }
        options = options || {};

        // Instantiate our own.
        var xmlHttpReq = $.ajaxSettings.xhr();

        // Make it use our own.
        options.xhr = function () {
            return(xmlHttpReq);
        };

        var $newDeferred = $.Deferred();
```

```
        var $oldPromise = $originalAjax(url, options)
            .done(function done_wrapper( response, text_status, jqXHR) {
                return($newDeferred.resolveWith(this, arguments));
            })
            .fail(function fail_wrapper(jqXHR, text_status, error) {
                return($newDeferred.rejectWith( this, arguments));
            })
            .progress(function progress_wrapper() {
                window.console.warn("Whoa, jQuery started actually using deferred
progress to report Ajax progress!");
                return($newDeferred.notifyWith( this, arguments));
            });

        var $newPromise = $newDeferred.promise();

        // Extend our own.
        $newPromise.progress = function (handler) {
            // Download progress
            xmlHttpReq.addEventListener('progress', function
download_progress(evt) {
                // window.console.debug( "download_progress", evt );
                handler.apply(this, [evt]);
            }, false);

            // Upload progress
            xmlHttpReq.upload.addEventListener('progress', function
upload_progress(evt) {
                // window.console.debug( "upload_progress", evt );
                handler.apply(this, [evt]);
            }, false);

            return(this);
        };

        return($newPromise);
    };
})(window, jQuery);
```

edited May 17 '15 at 9:52
Flo-Schield-Bobby
2,959 • 1 • 24 • 43

answered Feb 3 '14 at 17:58
MarkMYoung
76 • 7

So I just tried implementing your solution but this code is a little bit too pro for me to understand - how do I use this? I copy pasted your whole code before my document.ready and tried doing `$.ajax({ ... }).progress(function(evl) { console.log(evl); });` but nothing is happening. Can you help me? :) – Patrick DaVader May 21 '15 at 10:57

Which version of jQuery are you using? – Flo-Schield-Bobby May 28 '15 at 13:20

Can u add a Fidddle??... – Universal Grasp Jun 26 '16 at 12:31

http://www.htmlgoodies.com/beyond/php/show-progress-report-for-long-running-php-scripts.html

I was searching for a similar solution and found this one use full.

```
var es;

function startTask() {
    es = new EventSource('yourphpfile.php');

//a message is received
es.addEventListener('message', function(e) {
    var result = JSON.parse( e.data );

    console.log(result.message);

    if(e.lastEventId == 'CLOSE') {
        console.log('closed');
        es.close();
        var pBar = document.getElementById('progressor');
        pBar.value = pBar.max; //max out the progress bar
    }
    else {

        console.log(response); //your progress bar action
    }
});

es.addEventListener('error', function(e) {
    console.log('error');
    es.close();
});
```

}

and your server outputs

```
header('Content-Type: text/event-stream');
// recommended to prevent caching of event data.
header('Cache-Control: no-cache');

function send_message($id, $message, $progress) {
    $d = array('message' => $message , 'progress' => $progress); //prepare json

    echo "id: $id" . PHP_EOL;
    echo "data: " . json_encode($d) . PHP_EOL;
    echo PHP_EOL;
```

```
   ob_flush();
   flush();
}


//LONG RUNNING TASK
 for($i = 1; $i <= 10; $i++) {
    send_message($i, 'on iteration ' . $i . ' of 10' , $i*10);

    sleep(1);
 }

send_message('CLOSE', 'Process complete');
```

edited Aug 16 '15 at 10:29

answered Aug 16 '15 at 8:12

Sri Nair

11 ● 6