# **Manejo de errores en jQuery.ajax() jQuery**

* [Juan Padial](https://cybmeta.com/sobre-mi)
* 14 julio, 2014 - Actualizado el 28 marzo, 2016

Publicado el

jQuery nos ofrece fácil acceso a solciitudes AJAX mediante varios métodos: $.get, $.post, $.ajax, $.getJSON y alguno que otro más. En todos ellos se pueden poner en cola varias ****funciones diferidas****. Por ejemplo, .done(), que se ejecuta cuándo la solicitud Ajax ha concluido satisfactoriamente (esto no quiere decir que el contenido de la respuesta sea satisfactorio), .fail(), que se ejecuta en que caso de que haya ocurrido algún error, u .always(), que se ejecuta en ambos casos:

$.ajax({

url: "test.html"

}).done( function() {

alert( 'Success!!' );

}).fail( function() {

alert( 'Error!!' );

}).always( function() {

alert( 'Always' );

});

En este tutorial vamos a ver que información se recibe en el callback .fail() y como la podemos utilizar para depurar una aplicación Ajax.

## .fail( callback )

Cuándo una solicitud Ajax realizada con jQuery falla, se ejecuta la función callback especificada en .fail(). En el callback se reciben tres argumentos:

$.ajax({

url: "test.html"

}).fail( function( jqXHR, textStatus, errorThrown ) {

alert( 'Error!!' );

});

Veamos cada uno de ellos:

* object jqXHR: es un [objeto jqXHR](https://api.jquery.com/Types/" \l "jqXHR) que contiene todos los datos de la solicitud Ajax realizada, incluyendo la propiedad jqXHR.status que contiene, entre otros posibles, el código de estado HTTP de la respuesta.
* string textStatus: texto que describe el tipo de error, que puede ser, además de null, «abort», «timeout», «No Transport» o «parseerror».
* string errorThrown: si hay un error HTTP, errorThrown contiene el texto de la cabecera HTTP de estado. Por ejemplo, para un error HTTP 404, errorThrown es «Not found»; para error un HTTP 500 es «Internal Server Error».

A través de estos tres argumentos tenemos acceso a todos los posibles errores de una solicitud Ajax realizada con jQuery, que pueden ser:

1. HTTP 404 Error – la URL especificada para la solicitud ajax no se encontró
2. HTTP 500 Error – error interno del servidor
3. AJAX JSON Parse Errors
4. jQuery AJAX Timeout Errors
5. jQuery AJAX Abort Errors
6. Errores de conectividad
7. AJAX no transport error
8. Otros errores desconocidos

Combinando todo, podríamos detectar cada uno de estos errores a través de .fail() del siguiente modo:

$.ajax({

url: "test.html"

}).fail( function( jqXHR, textStatus, errorThrown ) {

if (jqXHR.status === 0) {

alert('Not connect: Verify Network.');

} else if (jqXHR.status == 404) {

alert('Requested page not found [404]');

} else if (jqXHR.status == 500) {

alert('Internal Server Error [500].');

} else if (textStatus === 'parsererror') {

alert('Requested JSON parse failed.');

} else if (textStatus === 'timeout') {

alert('Time out error.');

} else if (textStatus === 'abort') {

alert('Ajax request aborted.');

} else {

alert('Uncaught Error: ' + jqXHR.responseText);

}

});

En una web o aplicación cualquiera es bastante normal que haya varias interacciones Ajax y en cada una tendríamos que ejecutar un .fail() similar al anterior. Para evitar esta repetición podemos utilizar jQuery.ajaxSetup() y el parámetro error que veremos a continuación.

## El método jQuery.ajaxSetup() y el parámetro error

A través del método jQuery.ajaxSetup() podemos establecer los parámetros por defecto que jQuery utilizará para todas las solicitudes ajax. La sintaxis general tiene esta forma:

jQuery.ajaxSetup({

param : value

});

El parámetro error permite definir una función callback que se ejecuta cuándo la solicitud Ajax falla y que recibe los mismos argumentos que .fail(). De este modo, podemos definir una función que se ejecute para todas las solicitudes Ajax en caso de fallo y que nos diga que ha pasado;

$.ajaxSetup({

error: function( jqXHR, textStatus, errorThrown ) {

if (jqXHR.status === 0) {

alert('Not connect: Verify Network.');

} else if (jqXHR.status == 404) {

alert('Requested page not found [404]');

} else if (jqXHR.status == 500) {

alert('Internal Server Error [500].');

} else if (textStatus === 'parsererror') {

alert('Requested JSON parse failed.');

} else if (textStatus === 'timeout') {

alert('Time out error.');

} else if (textStatus === 'abort') {

alert('Ajax request aborted.');

} else {

alert('Uncaught Error: ' + jqXHR.responseText);

}

}

});

Una vez establecido este error callback predeterminado, se ejecutará para todas las solicitudes ajax que realicemos sin tener que definir un .fail() específico para cada una. Por ejemplo, podríamos utilizar tan solo:

$.get("prueba.html");

Como ventaja añadida, la puerta queda abierta para especificar una función .fail() u otro callback para error si lo necesitamos para una solicitud Ajax específica:

$.ajax({

url: "test.html",

error: function( jqXHR, textStatus, errorThrown ) {

// Otro manejador error

}

}).fail( function( jqXHR, textStatus, errorThrown ) {

// Un callback .fail()

});

La utilidad de esta implementación es clara: no nos arreglará el error que podamos tener pero ****nos mostrará que tipo de error se ha producido****, primer e imprescindible paso para poder solucionarlo.

## Advertencia para solicitudes cross-domain

La diferencia entre el parámetro error de jQuery.ajax() y .fail() es que error es un [evento ajax](http://api.jquery.com/Ajax_Events/) y .fail() es una [función diferida](http://api.jquery.com/deferred.fail/). Por tanto, la función callback definida en error no se ejecutará para solicitudes cross-domain pero si lo hará la definida en .fail().