5 New jQuery Ajax() Examples jQuery 1.9+

Facebook Twitter Was this helpful?

Into 2013 … And the way we use the jQuery.Ajax() function has changed over recent versions. With this in mind, and since jQuery 1.9 only supports (without I have coded up **5 New jQuery.ajax() Examples jQuery 1.9+** to show how you might use Ajax with newer versions of jQuery 1.9.x and 2.0.

The new way has some advantages over the old way.

I'll try keep this post updated with skeleton Ajax snippets for reference. As always comments welcome.

A quick reminder for those learning:

- The done() method replaces the deprecated jqXHR.success() method.
- The fail() method replaces the deprecated jqXHR.error() method.
- The always() method replaces the deprecated jqXHR.complete() method.

jQuery 1.9+ Ajax Example 1 – Subscribe to newsletter

This example shows a basic Ajax request for subscribing to a newsletter to send name and email to a backend script.

```
var subscribeRequest = $.ajax({
    type: "POST",
    url: "subscribe.php",
    data : { subscribeName: $('#name').val(), emailAddress: $('#email').val() }
});

subscribeRequest.done(function(msg,

    alert("You have successfully subscribed to our mailing list.");

);

subscribeRequest.fail(function(jqXHR, textStatus:

    alert("We could not subscribe you please try again or contact us if the prob

);
```

jQuery 1.9+ Ajax Example 2 – Request Timeout

This example shows how you might catch errors and fails such as timeouts for an ajax request.

> *Set a timeout (in milliseconds) for the request. The timeout period starts at the point the A.jax call is made; if several other requests are in progress and the browser has no connections available, it is possible for a request to time out before it can be sent. In Firefox 3.0b5 only, script and JSONP requests cannot be cancelled by a timeout; the script will run even if it arrives after the timeout period.*

```
var modalRequest = $.ajax(
    url: "getNewsData.php",
    timeout: 1000, // request after 10 seconds
    data : {timestamp: new Date().getTime() }
);

modalRequest.done(function(data:

    console.log(data);

);

modalRequest.fail(function(jqXHR, textStatus:

    if (jqXHR.status === 0:

        alert('Not connect.\n 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 (exception === 'parsererror':

        alert('Requested JSON parse failed.');

    else if (exception === 'timeout':

        alert('Time out error.');

    else if (exception === 'abort':

        alert('Ajax request aborted.');

    else:

        alert('Uncaught Error.n' + jqXHR.responseText:

);
```

jQuery 1.9+ Ajax Example 3 – Datafilter

This example shows how you can use the dataFilter function to process the raw data which is returned by your Ajax request.

```
var fileStateRequest = $.ajax(
    url : "getdata.php",
    dataFilter: function (data, type:

        //include any conditions to filter data here...
        //some examples below:

        //log - remove all comments from returned data
        return data.replace("/*", "");

        //log2 - if data is json process it in some way
        if (type === "json":
            {
                var parsed_data = JSON.parse(data:
                $.each(parsed_data, function(i, item:
                    {
                        //process the json data
                    }
                );
                return JSON.stringify(parsed_data:
            }
        else:
            {
                return data;
            }
    }
);

fileStateRequest.done(function(data:

    console.log(data);

);

fileStateRequest.fail(function(jqXHR, textStatus:

    console.log("Ajax request failed.. (" + textStatus + " - " + jqXHR.response

);
```

jQuery 1.9+ Ajax Example 4 – MIME Type

This example shows how to specify the content-response header type available on a XMLHttpRequest.

If you explicitly pass in a content-type to $.ajax(), then it'll always be sent to the server (even if no data is sent). If no charset is specified, data will be transmitted to the server using the server's default charset; you must decode this appropriately on the server side.

```
var contentTypeRequest = $.ajax(
    url : "getdata.php",
    contentType : "application/x-www-form-urlencoded; charset=UTF-8" //default

    //or choose from one below
    contentType : "application/atom+xml" //Atom
    contentType : "text/csv" //CSS
    contentType : "application/ecmascript" //JavaScript
    contentType : "image/png" //PNG Image
    contentType : "application/json" //JSON
    contentType : "application/pdf" //PDF
    contentType : "application/rss+xml; charset=ISO-8859-1" //RSS
    contentType : "application/x-tar" //Tarball
    contentType : "text/xml" //XML
);

contentTypeRequest.done(function(data:

    console.log(data);

);

contentTypeRequest.fail(function(jqXHR, textStatus:

    console.log("Ajax request failed.. (" + textStatus + " - " + jqXHR.response

);


//set specific accept headers (should work cross browser)
$.ajax(
    headers :
        Accept : "text/plain; charset=UTF-8",
        "Content-Type": "text/plain; charset=UTF-8"

);

//an alternative to the above
$.ajax(
    beforeSend: function(xhrObj:

        xhrObj.setRequestHeader("Content-Type","application/json");
        xhrObj.setRequestHeader("Accept","application/json");

);
```

jQuery 1.9+ Ajax Example 5 – Parsing XML

This example i grabbed from the jQuery docs Specifying the data type for ajax requests. It shows loading XML returned from a script and parsing the data as XML (if received by Internet Explorer as plain text, not text/xml).

By specify the $.ajax dataType option to be "text", make sure your server sends content with the "text/xml" MIME type. Sending the wrong MIME type will prohibit jQuery from correctly managing the data returned in the response, and could cause unexpected problems in your script.

```
var getXMLRequest = $.ajax(
    url : "getxmldata.php",
    contentType : "text/xml"
);

getXMLRequest.done(function(data:

    console.log(data);
    var obj;
    if (typeof data == "string":
        {
            obj = new ActiveXObject("Microsoft.XMLDOM");
            obj.async = false;
            obj.loadXML(data);
        }
    else
        {
            obj = data;
        }

    // Returned data available in object "obj"

);

getXMLRequest.fail(function(jqXHR, textStatus:

    console.log("Ajax request failed.. (" + textStatus + " - " + jqXHR.response

);
```

Was this helpful?

Get our latest JavaScript articles in your inbox, once a week, for free.

Enter your email
Get Updates

How do you make sure that the charting library you choose has everything you need? Sign up to receive this detailed guide from FusionCharts, which explores all the factors you need to consider before making the decision.

Email Me The Guide

COURSES >

JavaScript: Next Steps
Kyle Simpson
★★★★☆

React The ES6 Way
Dave Ceddia
★★★★☆

Your First Meteor 1.2 Application
Carl Vitullo
★★★★☆

BOOKS >

ECMAScript 2015: A SitePoint Anthology
Various Contributors
★★★★☆

Full Stack JavaScript Development with MEAN
Colin Ihrig
★★★★☆

JavaScript: Novice to Ninja
Darwin Dimitrov
★★★★☆

SCREENCASTS >

Building Realtime Web Apps with Firebase
Thomas Greco

Using Helpers and Conditionals in Ember
Thomas Greco

Modifying a Bootstrap Theme with Handlebars
Thomas Greco

About
Our Story
Advertise
Contact Us
Resources
Terms of Use
Privacy Policy
FAQ
Contact Us
CodeBase

Visit
SitePoint Home
Forums
Premium
Premium
Resources
Blog
Webinars

Connect

© 2008 - 2016 SitePoint Pty. Ltd.