jQuery: is plain text a valid alternative for handling AJAX responses?

JSON, HTML and XML are the most widely used data formats when using AJAX with jQuery. The only problem with this kind of content types is that they require some processing when you want to extract specific parts from the response. A feasible alternative is to use plain text. Really?

Simple responses

When you have to deal with simple responses such as status or transaction messages, the `text/plain` format is the most suitable. Bear in mind that jQuery is able to handle strings as HTML data so that you can use this format also for creating DOM structures.

Typically a server-side script handles plain text as follows:

```
header('Content-Type: text/plain');
// processes the request

echo 'Message';
exit();
```

In jQuery the AJAX request is pretty simple:

```
$.ajax({
```

```
        url: 'ajax.php',
        type: 'POST',
        dataType: 'text',
        data: {
                foo: 'Test'
        },
        success: function(text) {
                $('#output').text(text);

        }
});
```

You can also specify an HTML string on the server side:

```
header('Content-Type: text/plain');
// processes the request

echo '<p>Message</p>';
exit();
```

which you can process as HTML with jQuery:

```
$.ajax({
        url: 'ajax.php',
        type: 'POST',
        dataType: 'text',
        data: {
                foo: 'Test'
        },
```

```
        success: function(text) {
                $('#output').html(text); // as HTML
string

        }
});
```

Complex responses

When responses get more complex you have to perform some further steps both on the server and the client side. Suppose that you have to return validation messages after a form submission.

On the server-side script you have to build up a well-structured response string:

```
header('Content-Type: text/plain');
$errors = array();
$output = '';
// validates the form

if(count($errors) > 0) {
        foreach($errors as $error => $type) {
                $output .= 'error-' . $error . ':' .
$type . ',';

        }

} else {
        $output = 'Success';

}
```

```
echo $output;
exit();
```

If there are validation errors, the returned output string will be similar to the following:

`error-email:Not a valid e-mail address,error-username:Invalid username,`

With jQuery you have to manipulate and process the textual string:

```
$.ajax({
        url: 'ajax.php',
        type: 'POST',
        dataType: 'text',
        data: $('form').serialize(),
        success: function(text) {
            var html = '';
                // any errors?

                if(text.indexOf('error') != -1) {
                    // get rid of the last comma
                        var responseStr =
text.replace(/,$/, '');
                        var responseParts =
response.split(',');

                        for(var i = 0; i <
responseParts.length; i++) {
                                var responsePart =
responseParts[i];
                                var messages =
```

```
responsePart.split(':');
                                    var errorClasses =
messages[0].split('-');
                                    html += '<div
class="' + errorClasses[0] + ' ' + errorClasses[1] +
'">' + messages[1] + '</div>';

                            }

                } else {
                            html = '<p class="success">'
+ text + '</p>';

                }

                $('#output').html(html);
        }
});
```

Conclusions

As you can see, plain text is useful only for handling simple, atomic responses. When it comes to more complex responses, this kind of format requires a lot of string processing on the client side.