**COMPARISON OF VARIOUS AJAX TECHNIQUES**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Technique | Cross domain requests | Code intent and clarity | Error handling | History of Requests | REQUEST TYPE | Other points |
| 1. | Image based | YES | GOOD | GOOD | NO | GET only. | See \*\* below |
| 2. | Hidden Frames | NO and YES\* | AVG | POOR | YES | GET and POST |  |
| 3. | XHR based | NO and YES\* | VERY GOOD@ | VERY GOOD | NOβ | GET and POST |  |
| 4. | Script based | YES | GOOD# | GOOD$ | NO | GET only. |  |

\* - This needs to accomplished by using the “Access-control-allow-origin” header field in the server script that is supposed to allow the request to go through. Otherwise cross-domain calls are not possible with XHR.

# - This technique can only be used to dynamically load scripts at run time. When bandwidth is a constraint, then scripts may also be downloaded on a “need-basis”.

$ - onload and onerror events are available with script downloads also. Note that with script based AJAX, the script element should actually be added to the DOM to trigger the AJAX call. With image based AJAX, the image need not be part of the DOM. Merely changing the “src” attribute is enough.

@ - XHR can be used to download XML (as a DOM), JSON, text, video and images. It is the most versatile technique. Asynchronous file uploads are also easily accomplished using the “FormData” constructor.

\*\* - Can be very good when bandwidth constraints are severe. However, if cookies are disabled on the browser, this method can cause a lot of headache. If images are disabled on the browser, this method fails completely. Fortunately with today’s network speeds, images are rarely disabled. If we want to get plain text as a response, then cookies are the only way. We have to then read document.cookie on the client side, split on “;” and then split on “=” to read each cookie. Since these are part of the response header, we can only send limited text.

β – History can be maintained here too (by writing code), but not straight-forward like the Hidden Frames technique where nothing needs to be done. We have to use window.history.pushState() to save state and then use window.onpopstate event (which fires when we click the back or front buttons in the browser)