R. Nelson
12 October 1998

Category: EXPERIMENTAL

HTML REFRESH LANGUAGE (HTMLR/1.0)
<draft-rfced-exp-nelson-00.txt>

Abstract

This document describes HTML REFRESH, an EXPERIMENTAL language and protocol for refreshing HTML pages and allowing serious thin-client/server applications via HTTP [RFC2068].

1. Rationale and Scope

HTML forms have changed little in functionality or feature since the inception of the HTML standard. Whilst HTML forms allow the submission of form data from visible and hidden fields up to a server side CGI program (or some derivative thereof), the results must come back as a complete HTML page, either in the existing window/frame or in another browser window or frame.

This is particularly tedious as the entire target page needs to be redrawn, even if only certain data elements have been changed. This has two very negative affects. Firstly, the bandwidth requirements are increased as the entire page format must be sent down to the browser again and not just the "field" data which has changed. Secondly, the affect of redrawing the entire screen does not allow the development of user friendly thin-client/server applications (where the client is the web browser)and currently leads to user disorientation.

Various browser "add-ins", such as "Java" have been developed whilst HTML forms have largely been allowed to languish. This is extremely unfortunate as by far the largest number of transactions over the Internet occur via HTML forms.

This document specifies a HTML REFRESH language, which permits the refreshing of the form data elements and images on a HTML browser page without the redrawing of the entire page. This allows serious user interfaces to be developed whilst using less bandwidth to do so.

Future versions of this protocol may include extensions for refreshing non-form elements of a web page, in-line with DHTML standards.

2. HTML REFRESH LANGUAGE

The HTMLR language is built using the concepts of the HTML language and is to be used in web browsers in conjuction with HTML. Needless to say the main delivery method for HTMLR is HTTP, with the use of a new mime-type.

2.1 HTTP Added mime-type

The HTTP would allow the following mime-type through to the browser and the web server and browser would comprehend it. The mime-type is :

```
             text/htmlr
```

which would denote the content which followed as a HTML refresh. A HTML REFRESH aware browser would acknowledge the mime-type and note not to redraw the target page from scratch but instead integrate the results with it.

2.2 HTMLR Language

The HTMLR Language uses HTML like syntax to denote the refreshes that are to be made to a HTML page. The following tags and attributes are used to specify these refreshes. Each tag is covered below with accompanying description and example.

It is anticipated that HTMLR response pages would be generated by existing CGI (or like) capable programming languages, for example PERL, ASP, COLD FUSION, etc. Such languages should be easily capable of generating HTMLR and also changing the response mime-type.

2.3 HTMLR TAGS

2.3.1 HTMLR

Syntax:

```
<HTMLR> ... </HTMLR>
```

Description:
The HTMLR tag denotes that the all tags and text until the /HTMLR tag comprise a refresh of the existing HTML page/frame as displayed by the browser. This tag is equivalent in import to the <HTML></HTML> tags. Upon encountering a HTMLR tag, a browser should not clear the existing HTML display page/frame, but rather interpret the contents of the HTMLR tag and apply the relevant processing to the current page.

Valid tags within HTMLR tags are specified in the rest of this section.

Example:

```
<HTMLR>
     ..... refresh tags ....
</HTMLR>
```

2.3.2 WITHFORM

Syntax:

```
<WITHFORM  NAME="form-name">....</WITHFORM>
```

Description:
The WITHFORM tag denotes which form the tags within it apply to. The form-name specified with the NAME parameter must match the name of an existing form on the currently displayed page. The browser should treat all tags encountered within the WITHFORM screen as dealing with the specified form where applicable.

Tags which are affected by the WITHFORM tag are SETINPUT, SETTEXTAREA,CLEARINPUT,WITHSELECT.

If WITHFORM does not enclose these tags, they are deemed to be relating to the first form on the current page.

Example:

```
     <HTMLR>
             <WITHFORM NAME="Person">
             .... refresh tags for person form .....
             </WITHFORM>
     </HTMLR>
```

2.3.3 CLEARINPUT

Syntax:

```
<CLEARINPUT {EMPTY|DEFAULT}>
```

Description:
The CLEARINPUT tag clears all fields/checkboxes/radiobuttons/ textareas/buttons in the currently targeted form and resets them to either empty or their default values. The targeted form is the one specified in the enclosing WITHFORM tag, or in the absence of this, the first form on the page. This should be processed in sequence by the browser, thus any subsequent SETINPUT tags would set the fields away from their default or empty values.

The EMPTY attribute sets the fields to empty whilst the DEFAULT attribute set the fields to the original default value as specified in the original HTML page.

Example:

```
<HTMLR>
     <WITHFORM NAME="PERSON">
             <CLEARINPUT EMPTY>
     </WITHFORM>
     <STATUS VALUE="Person Record Added">
</HTMLR>
```

2.3.4 SETINPUT

Syntax:

```
<SETINPUT NAME="field-name" VALUE="new-value" {CHECKED|UNCHECKED}
     {DISABLED|ENABLED}>
```

Description:
The SETINPUT tag sets the input-field to the new-value specified in the VALUE parameter. For radio button and checkbox fields, the CHECKED/UNCHECKED parameter can be specified to alter the field appearance. The field-name, specified in the NAME parameter must match the name of a field (hidden/text/radio/checkbox/button) in the targeted form on the current page. The targeted form is the one specified in the enclosing WITHFORM tag, or in the absence of this, the first form on the page. For radio button fields, the new-value must also match the existing value of the named field in the current form.

The HTML 3.0 proposed (but not widely implemented) DISABLED parameter could also be used in SETINPUT, along with ENABLED to dynamically enable/disable the input field.

Example:

```
<HTMLR>
     <WITHFORM NAME="Person">
     <SETINPUT NAME="Name" VALUE="Fred Jones">
     <SETINPUT NAME="Dob" VALUE="26/Jan/1971">
     <SETINPUT NAME="Address" VALUE="35 Fred Street, Springfield">
     <SETINPUT NAME="Sex" Value="MALE" CHECKED>
     </WITHFORM>
</HTMLR>
```

2.3.5 SETTEXTAREA

Syntax:

```
<SETTEXTAREA NAME="field-name" {ENABLED|DISABLED}
                     >new-value</SETTEXTAREA>
```

Description:
The SETTEXTAREA tag sets the input-field to the new-value
specified before the closing /TEXTAREA tag. The field-name,
specified in the NAME parameter must match the name
of a textarea field in the targeted form on the current page.
The targeted form is the one specified in the enclosing WITHFORM
tag, or in the absence of this, the first form on the page.

The HTML 3.0 proposed (but not implemented) DISABLED parameter
could also be used in SETTEXTAREA, along with ENABLED to
dynamically enable/disable the textarea.

Example:

```
<HTMLR>
     <WITHFORM NAME="Person">
             <SETTEXTAREA NAME="Comments">
             The comments for this record are these
             </SETTEXTAREA>
     </WITHFORM>
</HTMLR>
```

2.3.6 SETFOCUS

Syntax:

```
<SETFOCUS FORM="form-name" FIELD="field-name">
```

Description:
The SETFOCUS tag set the input focus the field/textarea/selectlist
/checkbox/radiobutton-set/button with the name specified by the FIELD
parameter. The form the field is in is specified by the FORM
parameter. This tag is not affected by the WITHFORM tag as it
must set a definitive focus for the entire page, regardless of
how many forms are present.

Example:

```
<HTMLR>
     <MSGBOX>You must enter an Name</MSGBOX>
     <SETFOCUS FORM="Person" FIELD="Name">
</HTMLR>
```

2.3.7 WITHSELECT

Syntax:

```
<WITHSELECT NAME="field-name"  {DESELECTALL} {REMOVEALL}
             {ENABLED|DISABLED}></WITHSELECT>
```

Description:
The WITHSELECT tag is used to choose and set a select list object in the current form. The field-name, specified in the NAME parameter must match the name of a select list object in the targeted form on the current page. The targeted form is the one specified in the enclosing WITHFORM tag, or in the absence of this, the first form on the page.

The DESELECTALL parameter immediately de-selects all existing items in the select list. The REMOVEALL parameter immediately removes all items from the select list.

The HTML 3.0 proposed (but seldom implemented) DISABLED parameter could also be used in WITHSELECT, along with ENABLED to dynamically enable/disable the SELECT list.

Example:

```
<HTMLR>
<WITHSELECT NAME="Continent" CLEARALL ENABLED>
     <SETOPTION SELECTED>Asia</A>
</WITHSELECT>
</HTMLR>
```

2.3.8 SETOPTION

Syntax:

```
<SETOPTION {ADD|DELETE} SELECTED|DESELECTED
      VALUE="return-value">display-value</OPTION>
```

Description:
The SETOPTION tag is used to add, alter, or delete a select list item of the current SELECT list object. The current select list is the select list named by the last WITHSELECT within the currently targeted form. The SETOPTION tag is invalid outside of a WITHSELECT. The targeted form is the one specified in the enclosing WITHFORM tag, or in the absence of this, the first form on the page.

The ADD/DELETE parameter is used to add and delete items respectively from the SELECT list. The SELECTED/DESELECTED parameter is used to select/deselect an item after it has been created, or if it already exists, to alter it.

Example:
See WITHSELECT tag example

2.3.9 MSGBOX

Syntax:
```
<MSGBOX {TITLE="title"}>message</MSGBOX>
```

Description:
The MSGBOX tag displays a centered message box to the user with message supplied before the </MSGBOX> parameter enclosed in it. The message box must be modal and have an 'OK' button to allow the user to proceed. The browser should process the MSGBOX tag immediately before parsing/processing any more of the HTMLREFRESH. The optional TITLE parameter specfies a title for the messagebox window.

The text between MSGBOX and /MSGBOX tags should not contain HTML formating and browsers may wrap the text as well as obey CRLF combinations found in the text.

The MSGBOX tag allows for easy server generated intrusive messages without affecting the browser page display.

Example:
```
<HTMLR>
     <MSGBOX TITLE="Update Successful"
             >The Record has been updated.</MSGBOX>
</HTMLR>
```

2.3.10 STATUS

Syntax:
```
<STATUS VALUE="status-line-value">
```

Description:
The STATUS tag is used to place the value specified in the VALUE parameter into the status line at the bottom of the browser window.

The STATUS tag allows for another form of easy server generated intrusive messages without affecting the browser page display.

Example:
```
<HTMLR>
     <STATUS VALUE="Please correct the value in the Age Field.">
     <BELL>
</HTMLR>
```

2.3.11 PRINT and PRINTURL

Syntax:
```
<PRINT {TO=printer-name} {ORIENT=orientation} {TRAY=traynumber}
        {COPIES=copy-count}>....</PRINT>
<PRINTURL {TO=printer-name} {ORIENT=orientation}
      {TRAY=traynumber} {COPIES=copy-count} SRC="url">
```

Description:
The PRINT tag is used to print HTML to the specified printer. The HTML to print is supplied between the PRINT and /PRINT tags. The print is sent to the printer specified by the optional TO parameter. If no TO parameter is specified, a printer dialog should be displayed for the user to select a target printer from. Printing should occur in parallel to any other browser processing. The TO option is of most value in an intranet environment.

The ORIENT, TRAY and COPIES parameters are all options which allow control over the printing process. The ORIENT parameter can be used to specify "landscape" or "portrait" printing. The TRAY parameter can be used to select a paper source. The COPIES parameter can be user specify an number of copies to print. All are optional and are most suited to intranet systems.

The PRINTURL tag functions the same as the PRINT tag in terms of parameters, except that the content to print is supplied by the url specified in the SRC parameter. The browser should open the specified url and print the resultant stream as requested. The printing method should be dictated by the mime-type returned.

Browsers should aim to support multiple PRINT requests in a single HTML REFRESH stream.

The HTML allowable between the PRINT and /PRINT tags should be of the same conformance level as the normal HTML supported by the browser and print exactly the same as a user activated print of a normal web page.

Example:
```
<HTMLR>
     <MSGBOX>The person record will now be printed to your
           "HP" printer.</MSGBOX>
     <PRINT TO="hp01" ORIENT="portrait" TRAY="3" COPIES="1">
             <HTML>
                     <HEAD>
                             <TITLE>Person Record 123321</TITLE>
                     </HEAD>
                     <BODY>
                             <H2>Person Record 123321</H2>
                             <B>Name:</B> John Smith<BR>
                             <B>DoB: </B>  14/Mar/1969<BR>
                             <B>Address: </B> 14 James St Smithville<BR>
                             <HR>
                     </BODY>
             </HTML>
     </PRINT>
</HTMLR>
```

2.3.12 BELL

Syntax:

```
<BELL>
```

Description:

The BELL tag makes the browser produce an audible or visible bell.

Example:

```
<HTMLR>
     <BELL>
     <MSGBOX>The server has detected an error.</MSGBOX>
</HTMLR>
```

2.3.13 SETIMG

Syntax:

```
<SETIMG NAME="image-name" SRC="url">
```

Description:

The SETIMG tag is used to set images to new images based on a new URL. The "image-name" given in the NAME parameter must match the name of an image on the current HTML page. The new image is loaded into the same screen area as specified by the original IMG tag on the original HTML page.

The browser will place the new image on the page in the same location as the old image, with the same dimensions to avoid page resizing.

Example:

```
<HTMLR>
     <SETIMG NAME="EmployeePic" SRC="/images/employee/002012.jpg">
</HTMLR>
```

3. Operational Constraints and Implications

3.1 Web Servers

Web servers may require configuration to allow the text/htmlr mime-type to be transmitted from the CGI program.

3.2 Web Browsers

Web browsers will naturally be required to support the protocol with substantial internal changes. On reciept of a HTML REFRESH of a given page, the page will not be redrawn but instead the fields altered as required. The refresh should NOT be placed in any history or "BACK" button cache as this does not make sense.

3.3 Javascript/VBscript Implications

Javascript/VBscript browser implementations could possibly be extended to support an "OnRefresh" event in a similar manner as the existing "OnLoad" event. This event would be triggered upon receipt and application of a HTML REFRESH to the page. Appropriate extensions to the HTML BODY tag syntax would need to be made to support the "OnRefresh".

3.4 CGI Programs

CGI program authors would gain the freedom to write serious thin-client/server applications with HTML REFRESH. For example, a HTML page could have buttons to move forward and backward though records in a database. Upon pressing either button, a submission would be sent to the appropriate Web Server/CGI program. It would navigate the the next/previous database row and return new data for the HTML form fields using a HTML REFRESH.

This refresh would only alter the values in the HTML FORM fields on the page, thus lessening bandwidth requirents, aiding usability and removing redundant page redraws.

3.5 Security

HTML REFRESH pages would travel under HTTPS the same as HTML and therefore enjoy the same security benefits.

4. Acknowledgements

Thanks in particular to Steve Aldred, Nigel Williams and last but not least Joanna Ladon for encouragement and review.

5. References

[RFC2068] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2068, January 1997.

6. Author's Address

Ross Nelson
Wizard Information Services
15 Barry Drive
TURNER ACT 2612
Australia

EMail: Ross.Nelson@wizardis.com.au