Editorial de la
Universidad Tecnológica Nacional – U.T.N.

Tutorial

1.- Página web con recarga (refresh) automática, ignorando la memoria cache del navegador (browser).

En los comienzos de Internet las conexiones a los servidores eran lentas, lo que significaba que la carga y presentación en pantalla de las páginas web tardaba un tiempo (mucho comparado con las conexiones y los equipos actuales).

Un truco que adoptaron los navegadores (browsers) fue guardar las páginas bajadas en una memoria temporal (memoria cache) de la propia PC del usuario.

De esta manera, cuando en otro momento el usuario volvía a solicitar que se baje alguna de esas páginas, el navegador no se conectaba al servidor web sino que extraía la página de la memoria temporal (cache) de su PC.

La gran ventaja (en aquellas épocas) era que el usuario tenía la sensación de que la página se había descargado rápidamente.

Pero la desventaja era que si la página había tenido algún cambio (actualización) en el servidor el usuario no se enteraba.

De hecho y para paliar parcialmente este problema, los navegadores incluían (e incluyen) algún procedimiento o ícono sobre el cual clickear que saltea la memoria cache para conectarse realmente al servidor web.

Este es el ícono típico en las barras de herramientas para esta función:

Curiosamente este método se sigue aplicando actualmente, cuando las conexiones y los equipos son muy rápidos. Subsiste así el inconveniente de la falta de actualización del cache, cosa que puede confundir al usuario que no se entera de la actualización real de la página que está en el servidor. *[Ver ejemplo básico al final en Anexo I]*

I.- Página web con recarga (refresh) automática

Es posible incluir en una página web una instrucción que obligue al navegador (browser) a *bajar la página directamente del servidor*.

Esto es práctico cuando la página tiene cambios frecuentes, por ejemplo una página informativa que se actualiza repetidamente o el índice de una biblioteca virtual a la que se le van agregando títulos de obras nuevas.

Para esto se recurre al comando de JavaScript ***location.reload*** incluyéndolo en ***<body….>*** (comienzo del contenido de la página):

<body onload="JavaScript:location.reload(true); break;">

Ejemplo del código de una página que incluye esta instrucción:

<html>

<head> [contenido del encabezado] </head>

<body onload="JavaScript:location.reload(true); break;">

[contenido de la página: textos, imágenes, etc.]

</body>

</html>

II.- Comentarios a la instrucción

<body onload="JavaScript:location.reload(true); break;">

[onload] Atributo HTML aplicado en este caso a la etiqueta (tag) *<body>*: indica al navegador que al finalizar la carga del contenido de la página, debe ponerse en marcha el comando JavaScript ***[location.reload].***
Su sintaxis es: <body onload="script a ejecutar">

Fuente: http://www.w3schools.com/TAGs/ev_onload.asp

Nota: el atributo *onload* puede utilizarse para realizar otras funciones

[location.reload] este comando le está diciendo al navegador que debe recargar (reload) la página solicitada (location).

[(true)] Este parámetro del comando (por eso va entre paréntesis) confirma que la recarga se debe realizar desde el servidor. El otro parámetro posible - pero sin usa práctico - es ***(false)***: indica recuperar la página desde la memoria cache.

[break] Este comando informa al navegador que termine *(break)* el proceso una vez recargada la página. Sin este comando se corre el riego de que nuevamente se vuelva a recargar la página, nuevamente se vuelva recargar la página, etc., y se termine en un loop inacabable.

Nota importante:

Aunque parezca a primera vista algo absurdo, este procedimiento de actualización funciona si la página en su versión inicial ya fue bajada en algún momento por el usuario y - en consecuencia - está en la memoria cache.
Esto es obvio porque cuando posteriormente el usuario vuelva a intentar bajar la página –que ahora en el servidor contiene una nueva versión modificada – la carga inicial desde el cache descubre el comando y lo ejecuta, recargando la página desde el servidor.

Conclusión práctica: el ideal es que las páginas que sufren cambios frecuentes tengan el comando desde su origen.

III.- Aspectos técnicos del comando *location.reload*

Syntaxis: ***location.reload*** *(parámetro)*

Parametro tipo *lógico (boolean)*

El parámetro especifica la forma de recarga:

true – Recarga la página desde el servidor [location.reload(true)]

false - Recarga la página desde la memoria cache. Es un parámetro redundante.

Anexo I

- **Ejemplo básico de las etapas de creación y uso de una página web.**

Primer caso: página web sin ninguna instrucción de recarga automática.

Etapa I: el titular del sitio web http://www.xxx.yyy.zz/ decide subir la página informativa info.html (versión 01).
Etapa II: un usuario pide al navegador de su PC que descargue la página http://www.xxx.yyy.zz/info.html
El navegador se conecta al servidor, baja la página y la muestra en la pantalla. A su vez la carga a la memoria cache de esa PC.
Etapa III: : el titular del sitio web http://www.xxx.yyy.zz/ decide actualizar página informativa info.html (ahora en su versión 02).
Etapa IV: el usuario anterior pide al navegador de su PC que descargue nuevamente la página http://www.xxx.yyy.zz/info.html. El navegador detecta que esa página está en la memoria cache, la saca y muestra en la pantalla. El usuario no se entera de que hay una versión 02.

Segundo caso: página web con instrucción de recarga automática.

Etapas I , II y III: igual que en el caso anterior.
Etapa IV: el usuario anterior pide al navegador de su PC que descargue nuevamente la página http://www.xxx.yyy.zz/info.html. El navegador detecta que esa página está en la memoria cache, la saca pero detecta la instrucción de recarga. Se conecta al servidor, la descarga (ahora es la versión 02) y muestra en la pantalla. El usuario ve la página actualizada correctamente.

Anexo II

Una variante para la codificación de la instrucción de recarga.

Se puede incluir el comando de JavaScript en el encabezamiento de la página <head>…</head> y convocarlo desde <body…>

En <head ..>

```
<script type="text/JavaScript">
        function AutoRecarga{
          location.reload();
}
</script >
```

En <body >

```
<body onload="JavaScript:AutoRecarga(true);">
```

Anexo III

Otra variante NO AUTOMATICA (recurriendo a un "botón" visible en la pantalla).

```
<body>
<button onclick="reCargar()">Recargar esta página</button>
        <script>
                function reCargar() {
                   location.reload();
                }
        </script>
[contenido de la página (textos, imágenes, etc.)]
</body>
```

2.- Cambios en un documento PDF instalado en la web

Todo documento (archivo) con formato PDF subido a la web *está relacionado con algún link* incorporado a una página HTML (o eventualmente incorporado a otro documento PDF).

Ejemplo:

Una página web que sirve de *INDICE* tiene (entre otros) un link con el nombre *"Introducción",* dirigido al archivo llamado *"intro.pdf"*

Cuando un usuario hace click sobre el link "Introducción", el programa navegador busca en la memoria interna de la PC (cache) el archivo "intro.pdf".
Caso 1.- Si lo encuentra (porque el usuario lo había linckeado previamente), lo muestra en la pantalla sin subir a la web.
Caso 2.- Si NO lo encuentra (porque el usuario nunca lo había linckeado previamente) sube a la web, lo baja y lo muestra en la pantalla.

¿Qué pasaría si el dueño del sitio web decide modificar posteriormente el contenido del archivo *"intro.pdf"*?

En el *Caso 1* el programa navegador recurre a la memoria cache y *muestra el archivo anterior* que NO estaba modificado. El usuario NO se entera de los cambios. Y esto ES UN PROBLEMA.

En el *Caso 2* todo anda bien, porque el navegador sube a la web y baja el archivo ya modificado.

¿Qué debería hacer el dueño del sitio para evitar el inconveniente del *Caso 1*?

Nota importante: *lo que sigue es válido si la página web que tiene el link de origen (en el ejemplo sería el link "Introducción") está preparada para saltear la memoria cache (ver la primera parte de este tutorial). En caso contrario lo que se indica a continuación podría no funcionar.*

a) Cambiar el nombre del archivo PDF modificado (por ejemplo *"intro-2.pdf"*).
b) Modificar el link para que se dirija al nuevo archivo.
c) Por las dudas (nunca está de más) mantener un archivo *"intro.pdf"* eliminando el texto original y avisando que hay otro archivo actualizado, incorporando un link a ese archivo modificado (en el ejemplo *"intro-2.pdf"*).
