Leyendo la respuesta del servidor

Ahora que ya hemos comprobado que se ha procesado la respuesta completamente (**readyState**) y que el servidor nos dio una respuesta correcta a la petición de nuestra página (**status**), podemos entonces trabajar con los datos obtenidos por la respuesta.

Estos **datos** están **almacenados** en la propiedad **responseText** del objeto XMLHttpRequest.

Detalles del formato en cómo obtenemos la respuesta no se estudiará. Simplemente podremos obtener la respuesta en el formato que deseemos: separados por comas u otro carácter, una cadena de texto, etc..

En este ejemplo obtenemos una respuesta con datos separados por |

Veamos el listado 15.

Listado 15. Gestionando la respuesta

```
function actualizarPagina () {
     if (request.readyState == 4) {
       if (request.status == 200) {
         var respuesta = request.responseText.split("|");
         document.getElementById("solicitud").value = respuesta[0];
         document.getElementById("direccion").innerHTML =
           respuesta[1].replace(/\n/g, "<br>");
       } else
         alert("Estado es " + request.status);
     }
   }
```

Véanse gestión de expresiones Regulares http://mundogeek.net/archivos/2004/07/29/javascript-expresiones-regulares/ para la instrucción replace:

```
replace()

/g active búsqueda global. Cuando se usa en el método replace(), especifica que tiene que reemplazar todos los caracteres \n por <br
```

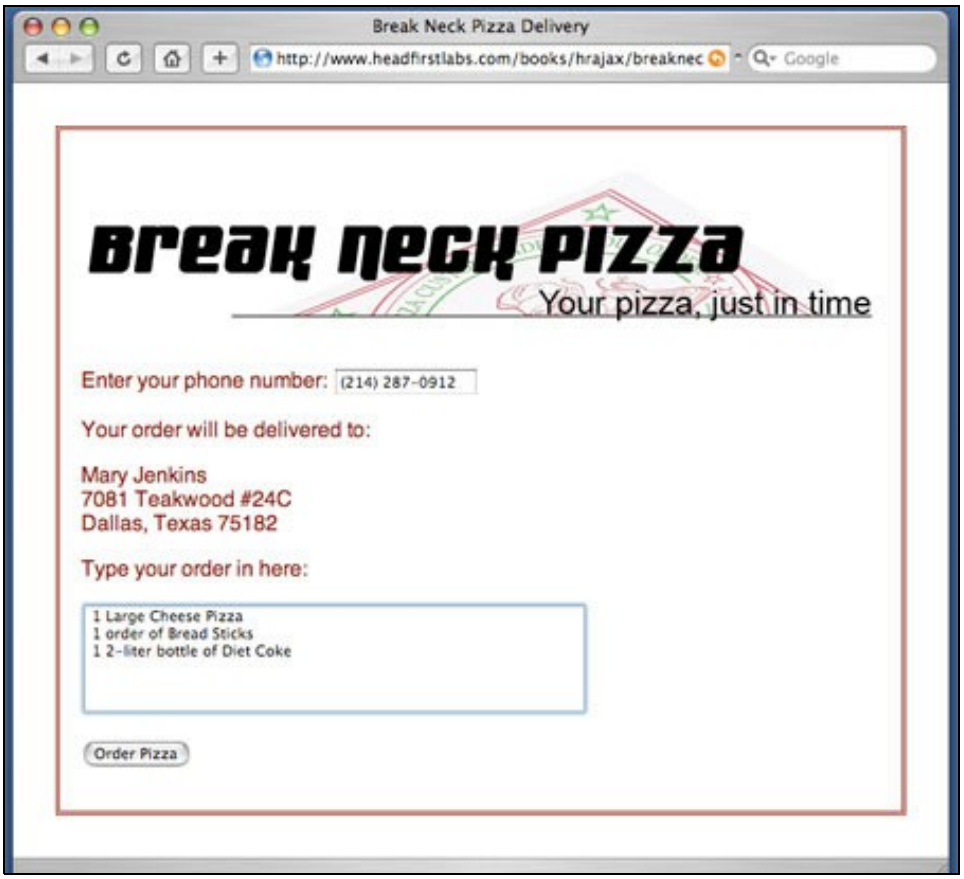
Primero **en responseText tendremos el total de la respuesta**. Como los datos vienen separados por | , mediante el método split los separamos y metemos en un array (respuesta). El array resultante contendrá en la primera posición (**respuesta[0]**) la última solicitud del cliente y en la segunda posición (**respuesta[1]**) la dirección del cliente. Como la dirección está en varias líneas, separadas por \n necesitamos convertir esos saltos de línea al carácter HTML de salto de línea
: esto lo hacemos con la sentencia replace().

Por último el texto modificado se coloca en nuestra página dentro del contenedor

y empleando la propiedad innerHTML. El resultado es que el formulario estará actualizado con la información del cliente como podemos ver en la figura 4.

Cada elemento HTML tiene una propiedad innerHTML que define tanto el codigo HTML como el texto que está entre la marca de apertura y de cierre. Cambiando la propiedad innerHTML podremos provocar mucha interactividad en las páginas. Tenemos que tener en cuenta que para poder usar innerHTML tenemos que dar a cada elemento de la página un identificador con la propiedad id. Mediante este identificador podremos acceder a ese objeto empleando la funcion getElementById(?elemento?) la cuál funciona en todos los navegadores.

Figura 4. Mostramos la dirección y debajo la última solicitud de nuestro cliente.



Deberemos mencionar además otra propiedad importante del objeto XMLHttpRequest llamada **responseXML**. Esta propiedad contiene (puedes adivinar?) una respuesta XML. Gestionar la respuesta XML es diferente a gestionar texto plano, así que incluye diferentes funciones que nos permitirán extraer ese contenido XML para introducir en nuestra página.