Ejercicios UD 10 AJAX

Ejercicio 1

A partir de la página web proporcionada, añadir el código JavaScript necesario para que:

1. Al cargar la página, el cuadro de texto debe mostrar por defecto la URL de la propia página.
2. Al pulsar el botón "Mostrar Contenidos", se debe descargar mediante peticiones AJAX el contenido correspondiente a la URL introducida por el usuario. El contenido de la respuesta recibida del servidor se debe mostrar en la zona de "Contenidos del archivo".
3. En la zona "Estados de la petición" se debe mostrar en todo momento el estado en el que se encuentra la petición (No inicializada, cargando, completada, etc.)
4. Mostrar el contenido de todas las cabeceras de la respuesta del servidor en la zona "Cabeceras HTTP de la respuesta del servidor".
5. Mostrar el código y texto de estado de la respuesta del servidor en la zona "Código de estado".

Ejercicio 2

La página HTML proporcionada incluye una zona llamada *ticker* en la que se deben mostrar noticias generadas por el servidor. Añadir el código JavaScript necesario para:

1. De forma periódica cada cierto tiempo (por ejemplo cada segundo) se realiza una petición al servidor mediante AJAX y se muestra el contenido de la respuesta en la zona reservada para las noticias.
2. Además del contenido enviado por el servidor, se debe mostrar la hora en la que se ha recibido la respuesta.
3. Cuando se pulse el botón "Detener", la aplicación detiene las peticiones periódicas al servidor. Si se vuelve a pulsar sobre ese botón, se reanudan las peticiones periódicas.
4. Añadir la lógica de los botones "Anterior" y "Siguiente", que detienen las peticiones al servidor y permiten mostrar los contenidos anteriores o posteriores al que se muestra en ese momento.
5. Cuando se recibe una respuesta del servidor, se resalta visualmente la zona llamada *ticker*.
6. Modificar la aplicación para que se reutilice continuamente el mismo objeto `XMLHttpRequest` para hacer las diferentes peticiones.

Ejercicio 3

Un ejemplo de validación compleja es la que consiste en comprobar si un nombre de usuario escogido está libre o ya lo utiliza otro usuario. Como es una validación que requiere el uso de una base de datos muy grande, no se puede realizar en el navegador del cliente. Utilizando las técnicas mostradas anteriormente y la página web que se proporciona:

1. Crear un script que compruebe con AJAX y la ayuda del servidor si el nombre escogido por el usuario está libre o no.
2. El script del servidor se llama `compruebaDisponibilidad.php` y el parámetro que contiene el nombre se llama `login`.
3. La respuesta del servidor es `"si"` o `"no"`, en función de si el nombre de usuario está libre y se puede utilizar o ya ha sido ocupado por otro usuario.
4. A partir de la respuesta del servidor, mostrar un mensaje al usuario indicando el resultado de la comprobación.

Ejercicio 4

Normalmente, cuando se valida la disponibilidad de un nombre de usuario, se muestra una lista de valores alternativos en el caso de que el nombre elegido no esté disponible.

Modificar el ejercicio de comprobación de disponibilidad de los nombres para que permita mostrar una serie de valores alternativos devueltos por el servidor.

El script del servidor se llama `compruebaDisponibilidadXML.php` y el parámetro que contiene el nombre se llama `login`. La respuesta del servidor es un documento XML con la siguiente estructura:

Si el nombre de usuario está libre:

```
<respuesta>
  <disponible>si</disponible>
</respuesta>
```

Si el nombre de usuario está ocupado:

```
<respuesta>
  <disponible>no</disponible>
  <alternativas>
    <login>...</login>
    <login>...</login>
    ...
    <login>...</login>
  </alternativas>
</respuesta>
```

Los nombres de usuario alternativos se deben mostrar en forma de lista de elementos (`<ul></ul>`).

Modificar la lista anterior para que muestre enlaces para cada uno de los nombres alternativos. Al pinchar sobre el enlace de un nombre alternativo, se copia en el cuadro de texto del login del usuario.

Ejercicio 5

Rehacer el ejercicio 14 para procesar respuestas del servidor en formato JSON. Los cambios producidos son:

1) El script del servidor se llama `compruebaDisponibilidadJSON.php` y el parámetro que contiene el nombre se llama `login`.

2) La respuesta del servidor es un objeto JSON con la siguiente estructura:

El nombre de usuario está libre:

```
{ disponible: "si" }
```

El nombre de usuario está ocupado:

```
{ disponible: "no", alternativas: ["...", "...", ..., "..."] }
```