EJERCICIOS AJAX

En los siguientes ejercicios se describen las funcionalidades básicas que debe realizar cada uno de ellos pero se valorará la inclusión de algunos elementos vistos en clase que no se piden en los ejercicios, como la utilización de webstorage, la modificación de clases css cuando se produzcan determinados eventos, ...

Ejercicio 1

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:

- Crear un script que compruebe con AJAX y la ayuda del servidor si el nombre escogido por el usuario está libre o no. (Ojo la respuesta es texto, por lo que habría que recuperarla con response.text())
- El script del servidor se llama compruebaDisponibilidad.php y el parámetro que contiene el nombre se llama login.
- 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.
- A partir de la respuesta del servidor, mostrar un mensaje al usuario indicando el resultado de la comprobación.

Ejercicio 2

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 (
).

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 3

Rehacer el ejercicio 2 para procesar respuestas del servidor en formato JSON. Los cambios producidos son:

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

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: ["...", "...", ..., "..."] }
```

Ejercicio 4

Crear un script que cargue de forma dinámica mediante AJAX la lista de provincias de un país y la lista de los municipios de cada provincia seleccionada.

1) Definir el código HTML de las dos listas desplegables vacías.

2) Cuando se cargue la página, cargar la lista de provincias en la primera lista desplegable. El script del servidor se llama cargaProvinciasXML.php. El formato de la respuesta es XML, con la siguiente estructura:

```
<provincias>
  <provincia>
    <codigo>01</codigo>
    <nombre>Álava</nombre>
  </provincia>
  ...
</provincias>
```
Para insertar las opciones en la lista desplegable, se pueden utilizar dos técnicas:

Propiedad innerHTML de la lista y código HTML de cada etiqueta <option>.

Crear elementos de tipo opción (new Option(nombre, valor)) y añadirlo al array options[] de la lista desplegable.

3) Añadir el evento adecuado a la lista de provincias para que cuando se seleccione una provincia, se carguen automáticamente todos sus municipios en la otra lista.

4) Cuando se seleccione una determinada provincia, se carga mediante AJAX la lista completa de municipios en la otra lista desplegable. El script del servidor se llama cargaMunicipiosXML.php. El parámetro que se debe enviar al servidor es el código de la provincia y el parámetro se llama provincia. El método que espera el servidor es POST. El formato de la respuesta es XML, con la siguiente estructura:

```
<municipios>
  <municipio>
    <codigo>0014</codigo>
    <nombre>Alegría-Dulantzi</nombre>
  </municipio>
  ...
</municipios>
```

Ejercicio 5

Modificar el ejercicio anterior para soportar las respuestas del servidor en formato JSON. Los cambios introducidos son los siguientes:

1) El script del servidor utilizado para cargar las provincias se llama cargaProvinciasJSON.php y la respuesta del servidor tiene el siguiente formato:

```
{ "01": "Álava/Araba",  "02": "Albacete",  "03":
"Alicante/Alacant", ...  }
```

2) El script del servidor utilizado para cargar los municipios se llama cargaMunicipiosJSON.php y la respuesta del servidor tiene el siguiente formato:

```
{ "0014": "Alegría-Dulantzi",  "0029": "Amurrio",  ...  }
```