Memoria Práctica 7

Interacción Persona-Computador

Integrantes del grupo:

SETIÉN PELAYO, LUIS

DESCALZO FERRER, VICTOR SHANTI

MONTENEGRO JARQUIN, DAVID EDMUNDO

ENTRECANALES CUENCA, OSCAR

---

1. Introducción:

En esta ocasión, desarrollamos un visualizador web para la tabla de compraventas del negocio de perfumería, el cual, mediante uso de JavaScript y jQuery, implementa funcionalidad interactiva de organización y personalización según se nos especifica.

2. Desarrollo de la interfaz web:

2.1. Aspectos generales:

Hemos creado una interfaz lo más simple posible de cara al usuario, anteponiendo delante de todas las opciones de personalizar el estilo de la página para que sea fácilmente visible desde un principio.
Destacamos la sencillez del diseño ya que el contenido no necesita de ser resaltado.
Por encima de todo hemos puesto un título sencillo y un favicon.
Se distinguen 2 containers principales: cambio de estilo y la tabla en sí.

2.2. Aspecto visual y estilo web:

Se estableció el diseño de la página utilizando Bootstrap para el ordenamiento de elementos y diseño básico de los mismos. Junto con el establecimiento de estilos CSS que permitan una mayor accesibilidad como el cambio de color oscureciéndose al pasar por los títulos de las columnas.

Se establecieron diferentes tamaños para los diferentes elementos como el título, los subtítulos y las cabeceras. Se hicieron divisiones entre elementos similares que tienen funciones diferentes. Asimismo, se hizo la posibilidad de que se ocultara la parte de personalización y volverla a mostrar con un clic en el título de la parte respectiva.

Mediante JavaScript se crearon distintas funciones basadas en colores que cambian partes visibles de la página según su descripción mediante la selección de un botón color para cada elección.

2.3. Visibilidad de las columnas:

Dentro del área “Personalización de Estilo”, he incluido una serie de etiquetas y *checkbox*, concretamente, uno por cada una de las columnas de la tabla, para que el usuario pueda seleccionar mediante los mismos las columnas que desea visualizar u ocultar. Para lograr que se cumpla su cometido al interactuar con ellos, el mecanismo interno se apoya sobre jQuery, y se resume en lo siguiente: cuando se detecte un cambio en alguno de los *checkbox*, éste desencadenará una llamada a la función *modificarColumna(indice)*, la cual se encarga de modificar la visibilidad del número de columna pasado como argumento. Si la columna se encontraba oculta (casilla de verificación previamente sin marcar), la hace visible mediante *tabla.show()*, y si se encontraba visible (casilla de verificación previamente marcada), se realiza *tabla.hide()*. La comprobación del valor actual se apoya en el uso de un array de booleanos, que modificamos y consultamos según el usuario va interactuando con la página. También es necesario aplicar un selector a la tabla para localizar la columna solicitada. Cabe mencionar que la funcionalidad resultante permite hacer cambios “al vuelo” sin tener que darle a un botón de “aplicar cambios” cada vez.

Un detalle a tener en cuenta es que, al refrescar o volver a visitar la página una segunda vez, es posible que el navegador haya almacenado en caché

las opciones previas que el usuario había indicado en los *checkbox*. Esto influye directamente en cómo debemos diseñar nuestro algoritmo. Hay varios enfoques para abordar este aspecto, yo he creído suficiente descartar cualquier opción en caché y resetear los valores de los *checkbox* en el momento en el que se cargue la web, por lo que indico mediante *window.onload* que ejecute una función que se encargue de dicha tarea.

2.4. Ordenación de la información en tabla:

El reordenamiento de las columnas en sentido ascendente y descendente fue un reto, ya que no solo hay que poder averiguar a que parte del header le hace click el usuario (para lo cual usamos .click de jQuery y $(this).index para distinguir entre cual columna basarnos para el ordenamiento), sino que también debemos encontrar una manera de modificar la tabla óptimamente, algo que a fuerza bruta requeriría una complejidad temporal y espacial demasiado alta para un tan simple comando. Para ello, hicimos uso de la funcion sort, y diseñamos una función de comparación para el caso de sentido ascendente (el cual es el valor por defecto) y descendente (al cual se llega despues de haberlo puesto en sentido ascendente), estas funciones comparan cada elemento de la columna indicada por el índice y deducen la posición de cada elemento dentro de esta y basandose en la función localeCompare(). La única diferencia entre el sentido ascendente y descendente es quien llama a esta última, ya que uno retornara un ordenamiento ascendente y el otro descendente después de ser organizado.

Tiene algunos problemas con números de varias cifras, ya que lee solo la primera cifra y lo ordena basandose en eso. No hemos podido hallar una solución para ese problema.

3. Reparto de trabajo:

A continuación, cada integrante del grupo indica la parte con la que ha contribuido personalmente a la resolución de la práctica propuesta:

- SETIEN PELAYO, LUIS:
    - Parte de la memoria e introducción de datos.

- DESCALZO FERRER, VICTOR SHANTI:
    - Sección 2.1. y 2.3. Aspectos generales y visibilidad de las columnas.

- MONTENEGRO JARQUIN, DAVID EDMUNDO:
    - Sección 2.2. Estilos css y elección de colores.

- ENTRECANALES CUENCA, OSCAR:
    - Sección 2.4. Ordenación de la información en la tabla.