U04P01: Farmacias de guardia

Objetivo

A partir del JSON proporcionado, crear una Web en la que se visualicen los datos que contiene dicho fichero.

El contenido del fichero JSON debe guardarse en un array de objetos. Para ello bastará con copiar el contenido del JSON y guardarlo en único array.

Concretamente tenemos un JSON con los datos de farmacias de guardia en Santa Cruz de Tenerife.

Requisitos

Los datos obtenidos del JSON se representarán (crearán, accederán, recorrerán,...) en el HTML utilizando métodos DOM con JS.

```
Elementos contenidos en el JSON
```

Cuando se muestre un elemento por pantalla se debe representar: su nombre, la dirección, su horario, su web y su teléfono.

- cada elemento debe representarse de manera diferenciada.
- se deben asociar varias clases que vendrán definidas en un ficheros CSS separado o un framework como Bootstrap a los distintos elementos anteriores.
- si no tiene web, no debe mostrarse nada (la palabra web, icono representativo de un navegador, www,....) asociado a la web.

```
Agrupamiento de la información de los elementos del JSON
```

Las farmacias deben mostrarse agrupadas por distrito y después por barrio.

Debemos mostrar de cada farmacia agrupada en un bloque en el que figure su nombre, la dirección, web y su teléfono. Si no tiene Web, no

Las farmacias deben mostrarse agrupadas por distrito y después por barrio.

El nombre del distrito debe mostrarse una única vez. Si hay varias farmacias en distintos barrios del mismo distrito, el nombre del distrito se mostrará una sola vez.

Si hay varias farmacias en el mismo barrio, el nombre del barrio se mostrará una sola vez.

Filtros

Inicialmente, al cargarse la página, deben mostrarse todos los elementos del JSON.

En la parte superior se deben disponer los siguientes filtros que crearemos usando JS. Se precisa al menos un filtro y un máximo de 3.

Se podrá filtrar por distrito y por barrio mediante algún elemento como por ejemplo, una caja de texto y su botón para buscar.

Cuando se realiza la búsqueda se deben crear los elementos dinámicamente mediante DOM y JS que cumplan con él o los criterios. Para ello, si ya hay elementos representados, debemos borrarlos mediante DOM y JS.

Entrega

- El proyecto debe tener la siguiente estructura de carpetas y ficheros:
  - U04P01_Apellidos_Nombre (carpeta del proyecto)
    - img (carpeta con las imágenes empleadas)
    - js (carpeta con los js creados)
    - css (carpeta con los ficheros css creados)
    - index.html (html que estará enlazado con un único fichero js minificado y ofuscado)
- Sube el proyecto comprimido en formato ZIP.
- El nombre del fichero debe seguir el siguiente formato:
  - U04P01_Apellidos_Nombre.zip
- Indenta adecuadamente el código y usa los comentarios que consideres oportunos.
- Elimina las importaciones de librerías que no uses.

Calificación

- Requisitos mínimos (70%):
  - Deberán estar implementadas todas las funcionalidades descritas.
  - Representación de la información:
    - uso óptimo de la propiedades DOM para la gestión a la hora de representar cada elemento: máx 0.75 ptos.
    - se cargan los elementos agrupados por distrito y por barrio (sin estar repitiendo el nombre del distrito ni el del barrio): máx 0.75 ptos.
    - se aplican clases diferentes a los elementos mediante DOM y JS: máx 0.5 ptos
    - al cargarse la web, se muestran por pantalla todos los elementos del JSON tal y como se indica en el enunciado: máx. 0.5 ptos.
  - Se borran todos los nodos y sus hijos al utilizar algún filtro: máx 0.5 ptos.
  - Los recorridos por el árbol DOM se realiza de manera eficiente: máx 0.5 ptos.
  - Se seleccionan los elementos utilizando cada uno de los siguientes métodos: getElementById, getElementsByTagName, querySelector y querySelectorAll: máx 0.5 ptos.
  - Por cada filtro (mínimo un filtro, máximo 3):
    - creación del filtro mediante JS y DOM: máx. 0.5 ptos.
    - se muestran correctamente los datos al aplicar el filtro creado mediante JS y DOM: máx 0.5 ptos.
- Buenas prácticas de programación (20%):
  - Sólo se valorará si se obtienen al menos 5 puntos de 7 en el apartado anterior.
  - Código mínimo imprescindible.

    - Código optimizado.
    - Comentarios necesarios y suficientes.
    - Variables y métodos con nombres significativos y autoexplicativos.
        - uso de ESLint (incluir al menos 3 capturas en las que se pueda distinguir que se usado): 0.5 ptos.
        - notación camelCase, funciones arrows, métodos map, filter, reduce,...: 0.5 ptos.
        - uso de JSDocs: 0.5 ptos.
        - código refactorizado: 0.5 ptos.
- Originalidad y creatividad (10%). Se valorarán positivamente las aportaciones creativas o innovadoras para enriquecer la aplicación:
    - Sólo se valorará si se obtienen al menos 7 puntos de 9 entre los dos apartados anteriores.
    - Se valorarán como máximo dos aportaciones.

El plagio o la incapacidad de defender la entrega se calificará con un cero.

Actividad propuesta: jueves, 7 de noviembre de 2019, 12:10
Entrega: miércoles, 13 de noviembre de 2019, 23:59