

TEMA 10 - ACTIVIDAD 2 (NO EVALUABLE). INTRODUCCIÓN A JQUERY

Desarrollo Web entorno cliente
CFGS DAW

Sergio García Barea
sergio.garcia@ceedcv.es
2018/2019

Versión:180717.1442


Licencia





Reconocimiento - NoComercial - CompartirIgual (by-nc-sa): No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

 Importante

 Atención

 Interesante

ÍNDICE DE CONTENIDO

1. Boletín de ejercicios.....	3
-------------------------------	---

UD010. INTRODUCCIÓN A JQUERY

1. BOLETÍN DE EJERCICIOS

1) Calcular números primos es una tarea costosa pero útil en diversos campos, sobre todo en la criptografía.

Tenemos en <http://hispabyte.net/DWEC/EjercicioUD10-1AJAX.php> un servidor que dado un número enviado por POST te dice si es primo.

Nosotros queremos saber si es primo y además palíndromo. El calculo de si es primo se hará en el servidor y el calculo de si es palíndromo en el cliente.

Completa el código dado para decir si un número es primo y palíndromo o si no lo es.

Ejemplos:

- 5 es primo y palíndromo
- 11 es primo y palíndromo
- 17 es primo y no palíndromo
- 4 no es primo pero si palíndromo
- 22 no es primo pero si palíndromo
- 14 no es primo ni palíndromo

2) Las APIs de Google Maps están disponibles en <https://developers.google.com/maps/>

En concreto ahora nos interesan las APIs de Google Street View <https://developers.google.com/maps/documentation/streetview/>

Queremos hacer un programa que cargue usando AJAX la foto de una localización indicada.

Esto se puede hacer con la llamada

<http://maps.googleapis.com/maps/api/streetview?size=600x300&location=miLocalizacion>

Donde miLocalizacion es la localización a buscar.

<http://maps.googleapis.com/maps/api/streetview?size=600x300&location=colon,valencia>

Haz una aplicación que pida la localización y actualice usando AJAX la imagen. Antes de comenzar planteate las siguientes preguntas:

- ¿Necesitamos el Ajax de jQuery?
- Con esa API de Google ¿Podemos usar simplemente la propiedad src de la imagen?

3) Investiga APIs que se puedan usar con AJAX de los principales sitios de Internet, compartiendo ejemplos proporcionado por terceros o tuyos propios.
¡Comparte lo encontrado usando el foro!

4) Realiza una web que tenga un botón “Obtener sopa de letras”. Cuando se pulse ese botón, obtendrá mediante AJAX una sopa de letras (en formato array de cadenas) y las palabras a buscar (también en un array de cadenas).

Al recibirla, cargará en pantalla la sopa de letras (mediante una tabla) y un combo-box donde estará el texto “Ninguna seleccionada” y después cada una de las palabras.

Al modificar ese combo-box se actualizarán los valores de un div con id=”resultado”. Si se elige “Ninguna seleccionada” el div indicará “No hay ninguna palabra seleccionada”.

Si se selecciona una palabra, indicará en que lugar de la sopa de letras está su primera letra y en cual su última, tomando las posiciones 0,0 como las iniciales.

Por ejemplo:

AASOPAB

XFGHZXX

Si buscamos SOPA, nos dirá posición inicial x=2,y=0 , posición final x=5,y=0.

Las palabras podrán estar en horizontal y en vertical y se leerán de izquierda a derecha (horizontal) o de arriba a abajo (vertical).

Recordad, que si en cualquier momento pulsamos el botón “Obtener sopa de letras”, se recargará dinámicamente la sopa de letras y sus palabras.

Tenéis un ejemplo de servidor de sopas de letas adjunto al enunciado y funcionando en <http://hispabyte.net/DWEC/EjercicioUD10-2AJAX.php>

5) Utilizando la API pública de Codeforces disponible en <http://codeforces.com/api/help>

Haz una aplicación que con el botón “Obtener problemas por TAG” y un input de tipo texto donde se guarde el TAG a pedir y un combo-box con los valores “ascendente” y “descendente” (por defecto ascendente).

Al pulsar el botón, mediante una petición AJAX obtendrá todos los problemas con ese el tag indicado en el input tipo texto y mostrará en la página los nombres de los problemas (únicamente ese valor) ordenado ascendente o descendente.

Si cambia el combo-box, los problemas se re-ordenaran dinámicamente (sin pedir la información por AJAX de nuevo).

Si se vuelve a pulsar el botón “Obtener problemas por TAG”, se volverá a hacer la petición AJAX y recargará el contenido de la página.

⚡ Repasando Ordenación

La ordenación se realizará o con las funciones que implementa Javascript https://www.w3schools.com/jsref/jsref_sort.asp y recomiendo que su uso este documentado en la chuleta del examen. No es obligatorio implementar métodos de ordenación “de cero” ni entender bien los complejos (Quicksort y Mergesort).

Comparación de algoritmos
<https://www.cs.usfca.edu/~galles/visualization/ComparisonSort.html>

Aquí algunos algoritmos divididos en “no óptimos” y “óptimos” y sus costes para tomarlos como referencia:

a) Método no óptimos

Tienen coste medio N^2 (es decir, si hay 100 elementos, el coste es $100 \cdot 100 = 10000$)

- Inserción https://es.wikipedia.org/wiki/Ordenamiento_por_inserci%C3%B3n
- Burbuja https://es.wikipedia.org/wiki/Ordenamiento_de_burbuja

b) Método mas óptimos

Tienen coste medio $N \cdot \log_2 N$. Es decir, si hay 100 elementos, el coste es $100 \cdot 6,643 = 664,3$)

6,643 es el log en base 2 de 100-.

- Quicksort <https://es.wikipedia.org/wiki/Quicksort>
- Mergesort https://es.wikipedia.org/wiki/Ordenamiento_por_mezcla

Implementación de QuickSort en Javascript:

- <https://antjanus.com/blog/web-development-tutorials/understanding-quicksort-js-native-implementation/>
- <http://hdeleon.net/algoritmo-de-ordenamiento-quicksort-en-javascript/>