Desarrollo de Aplicaciones telemáticas
Examen final, prueba de teoría. 26 de mayo de 2022
Grado en ingeniería telemática.
Universidad Rey Juan Carlos

Instrucciones:

- Ejecuta en un terminal `~mortuno/prepara`
- Esto dejará en el ordenador el fichero `~/dat.mayo.22/teoria.TULOGIN.txt`, donde debes escribir tus respuestas. TULOGIN será tu nombre de usuario en el laboratorio.

Ejercicio 1 (4 puntos)

Indica si los siguientes selectores CSS son correctos o no. Si son correctos, indica su significado. Si no son correctos, explica brevemente por qué. (Tal vez te ayude el considerar que esto podría aplicarse en un documento HTML donde tenemos elementos que representan personas y las vacunas que se han puesto)

1) .pfizer.azeneca
2) pfizer azeneca
3) #azeneca#pfizer
4) #pfizer azeneca

Respuesta

1. Correcto. Selecciona los elementos con las clases *pfizer* y *azeneca* simultáneamente.
2. Incorrecto. Seleccionaría los elementos HTML *azeneca* descendientes de elementos *pfizer*, pero esto no tiene sentido, en HTML no existen estos elementos.
3. Incorrecto. Sería un elementos con los identificadores *pfizer* y *azeneca* pero un elemento no puede tener dos identificadores distintos De la misma forma que, por ejemplo, no es legal que un español tenga dos DNI distintos.
4. Incorrecto. Sería un elemento HTML *azeneca* descendiente del elemento con id *pfizer*. Pero como hemos dicho, *azeneca* no es un elemento HTML.

Ejercicio 2 (2 puntos)

En *Bootstrap* las columnas (celdas) en ocasiones se muestran en disposición *normal* y en ocasiones en disposición *apilada*. Explica esto.

Respuesta

En *Bootstrap* y en todas las tecnologías web similares basadas en un *grid*, el contenido se coloca dentro de unos bloques llamados *columnas*, que nosotros preferimos llamar *celdas*. Las celdas se organizan en filas y columnas, las celdas de la misma fila está unas al lado de otras. Esta es la *disposición normal*, cuando la pantalla es lo bastante grande. Si la pantalla es menor, las celdas de una fila ya *no caben* unas al lado de otras, así que se colocan unas encima de otras, en *disposición apilada*.

Errores frecuentes

- Es un error hablar de *viewport virtual*. También es una técnica para adaptar el contenido a pantallas de tamaño variable, pero anticuada, de segunda generación. El *grid* es una técnica diferente, de tercera generación, que no utilizar *viewport virtual*.
- Es un error decir que las celdas se redimensionan o adaptan su tamaño, o usan más o menos columnas. No lo hacen. Lo que hacer es colocarse en disposición normal (horizontal) o apilada (vertical).

Ejercicio 3 (2 puntos)

Explica brevemente en qué consiste el *problema del código yo-yo*.

Respuesta

Es un inconvenient que se da en programación orientada a objetos basada en clases cuando hay jerarquías complejas, esto es: *clases padre*, *clases abuelo*, *clases bisabuelo*, *clases tatarabuelo*, etc.

Cuando un programador trabaja en una clase *nieto*, le afecta el código de todos sus *antepasados*, así que tiene que estar contiuamente subiendo y bajando por la jerarquía de clases, buscando las propiedades o los métodos relevantes en cada caso.

Observación importante: este es un problema de la programación orientada a objetos **solamente cuando las jerarquías son complejas**, lo cual no es especialmente habitual.

Ejercicio 4 (2 puntos)

En JavaScript ¿Qué es una *promesa*?

Respuesta

Es una técnica de programación concurrente. Es un objeto que inicialmente no tiene valor, pero que transcurrido cierto tiempo, si todo va bien, acabará teniendo valor. O indicará que se ha producido un error, si ha habido problemas.

Este objeto es devuelto por una función asíncrona, el código que llama a esta función se queda bloqueado esperando a que la promesa se resuelva.