**Aplicaciones Telemáticas.**
**Examen final, prueba de teoría. 16 de junio de 2026.**
Grado en Ingeniería Telemática.
Universidad Rey Juan Carlos.

---

### Instrucciones:

- Rellena el fichero `teoria.txt` con tus datos personales y la respuesta a las cuatro preguntas siguientes. No cambies el nombre del fichero.

# Ejercicio 1 (2.5 puntos)

Indica si los siguientes selectores CSS son correctos o no. Si son correctos, describe su funcionamiento. Si no son correctos, explica brevemente por qué. Un apartado erróneo o en blanco tendrá una penalización del 50 %. Dos o más apartados erróneos o en blanco, tendrán puntuación nula.

1) `div .span`

2) `div span`

3) `p  light`

4) `.light.vegano`

### Respuesta

1) Correcto. Selecciona los elementos de clase *span* descendentes de elementos *div*. No es recomendable usar *span* como nombre de clase porque es un elemento HTML e induce a confusión. Pero la norma lo permite.

2) Correcto. Seleccionaría los elementos *span* dentro de elementos *div*

3) Sintácticamente es correcto, aunque en la práctica no tiene sentido. Seleccionaría los elementos HTML de tipo *light* contenidos dentro de párrafos. Pero en HTML no hay ningún elemento estándar que se llame así.

4) Correcto. Se refiere a los elementos que tengan simultáneamente las clases *light* y *vegano*.

# Ejercicio 2 (2.5 puntos)

De cada una de estas afirmaciones, indica si es verdadera o falsa. Si es verdadera justifícala brevemente. Si es falsa, explica todos los motivos por los que lo es.

1. Con localStorage, la información es persistente. Pero no se puede almacenar en el disco del cliente, por motivos de seguridad esto está prohibido en JavaScript. Por tanto, la información se guarda en el servidor.

2. Si una aplicación almacena mucha información en sessionStorage, puede que quede poco sitio para las demás. Pero como el espacio disponible suele ser 5Mb, esto no es un gran inconveniente.

3. Si una aplicación almacena mucha información en localStorage, puede que quede poco sitio para las demás. Pero como el espacio disponible suele ser 5Mb, esto no es un gran inconveniente.

4. Con localStorage, la información es persistente, se guarda en el disco del cliente.

### Respuesta

1. Falso.

   Lo que está prohibido, porque sería un serio problema de seguridad, es que un programa JavaScript dentro del navegador accediera a cualquier fichero del disco del usuario. Con localStorage, la información se guarda en el disco del usuario, pero en un fichero controlado por el navegador. El programa JavaScript solo percibe el almacenamiento de pares clave-valor.

2. Falso. Cada aplicación dispone normalmente de unos 5Mb (aunque depende del navegador). Si usa mucho espacio, está consumiendo *sus* 5Mb. No afecta al resto de aplicaciones.

3. Falso.

   Por el mismo motivo que el apartado anterior.

4. Cierto.

   En el disco del cliente, aunque de forma controlada por el navegador, como hemos indicado en la primera respuesta.

## Ejercicio 3 (2.5 puntos)

En Bootstrap

1. ¿Qué es la *disposición normal*? ¿Cuándo se produce?

2. ¿Qué es la *disposición apilada*? ¿Cuándo se produce?

### Respuesta

1. La disposición normal es aquella en la que los elementos de una fila (row) aparecen uno al lado del otro, ocupando las columnas que se les han asignado mediante el sistema de rejilla de Bootstrap. Cada elemento conserva su ancho relativo dentro de la fila.

   Se produce cuando el ancho de la ventana es igual o superior al punto de ruptura (breakpoint) definido para esas columnas. En ese caso, Bootstrap distribuye los elementos horizontalmente.

2. La disposición apilada es aquella en la que los elementos que normalmente compartirían una misma fila pasan a mostrarse uno debajo de otro. Cada elemento ocupa todo el ancho disponible.

   Se produce cuando el ancho de la ventana es inferior al punto de ruptura especificado para las columnas. Bootstrap adapta así la presentación para facilitar la visualización en dispositivos con pantallas más pequeñas.

# Ejercicio 4 (2.5 puntos)

1. ¿Qué significa *Hora Unix*?. Describe brevemente su formato.
2. ¿La usa JavaScript?
3. ¿Qué significa *Hora UTC*?
4. ¿La usa JavaScript? ¿Siempre? ¿Nunca? ¿A veces? ¿Qué veces?
5. ¿Qué significa *Hora Zulú*?
6. ¿La usa JavaScript? ¿Siempre? ¿Nunca? ¿A veces? ¿Qué veces?

## Respuesta

1. Es el formato de hora típico de los sistemas Unix. Es un número entero, los segundos transcurridos desde el 1 de enero de 1970. (El *epoch*). No incluye directamente año, mes, día, hora, etc. Es necesario hacer una conversión para mostrarla en formato legible, pero facilita mucho hacer cálculos con fechas.

   No tiene en cuenta la hora local, esto es, siempre es hora UTC.

2. Los *intervalos* en JavaScript son algo muy parecido: los milisegundos transcurridos desde el 1 de enero de 1970.
3. *Coordinated Universal Time*. Es la hora universal, la hora del meridiano de Greenwich. No tiene en cuenta la zona del mundo.
4. Sí. Un objeto *Date* siempre contiene internamente una hora UTC. Y un intervalo (similar a la hora Unix pero en milisegundos) también es UTC. En otra ocasiones JavaScript usa hora local. Por ejemplo en el constructor de un *Date*.
5. *Zulú* es la palabra empleada en el alfabeto fonético OTAN para la letra *Z*, que a su vez en este contexto significa *zero*. Es la hora UTC+0, o lo que es lo mismo, la hora UTC.
6. Es aplicable todo lo dicho a la hora UTC, apartado 4.