

Actividades unidad 3

Actividades Teóricas

1. Cookies y almacenamiento web:

a. Explica qué es una cookie y para qué sirve en el desarrollo web.

Una cookie es un pequeño archivo de texto que un sitio web guarda en el navegador. Sirve para recordar información sobre los usuarios entre las diferentes páginas que visitas, como mantener la sesión iniciada, recordar qué hay en tu carrito de compra o tus preferencias de idioma.

b. Diferencia entre expires y max-age.

Expires usa una fecha definitiva mientras que max-Age define una fecha relativa a los segundos desde su creación.

c. ¿Qué tamaño máximo suele tener una cookie?

Son muy ligeras, generalmente tienen un límite de 4 KB por cookie.

d. Explica por qué es necesario usar `encodeURIComponent()` al guardar una cookie.

Principalmente porque las cookies no aceptan ciertos caracteres especiales como espacios, comas o punto y coma porque pueden romper el formato del archivo. Lo que hace esta función es adaptar el string a un formato válido.

e. ¿Cómo se borra una cookie?

Para borrarla, se debe sobrescribir. Creas una cookie con el mismo nombre pero le asignas una fecha de caducidad en el pasado o un max-age de 0.

2. localStorage y sessionStorage:

a. Diferencias entre localStorage y sessionStorage.

La principal diferencia es la duración. Los datos en localStorage persisten indefinidamente hasta que se borre. En cambio, sessionStorage se borra automáticamente en cuanto se cierra la pestaña o ventana del navegador.

b. ¿Qué métodos tiene localStorage para guardar y recuperar información?

Guardar: `localStorage.setItem('clave', 'valor')`

Recuperar: `localStorage.getItem('clave')`

c. Explica un caso real donde sería adecuado usar sessionStorage.

Es ideal para formularios de varios pasos (como un asistente de compra o registro) o datos bancarios sensibles. Si el usuario cierra la pestaña por error o termina la gestión, quieres que esa información temporal desaparezca por seguridad y limpieza, no que se quede guardada para siempre.

3. Manipulación de strings

a. ¿Qué diferencia hay entre `.substring()` y `.substr()`?

`.substring()` recorta basándose en los índices de inicio y final. `.substr()` recorta basándose en el inicio y la cantidad de caracteres a tomar

b. ¿Qué hace `.startsWith()` y qué devuelve?

Comprueba si un texto comienza con los caracteres indicados. Devuelve un valor booleano

c. ¿Qué diferencia existe entre `.replace()` y `.replaceAll()`?

`.replace()` solo cambia la primera coincidencia que encuentra. `.replaceAll()` cambia todas las apariciones que existan en el texto.

4. Objetos en JavaScript

a. Explica dos formas de crear objetos en JavaScript.

Usando llaves `{ clave: valor }` o usando el constructor `new Object()`.

b. ¿Qué hace el operador `in` en un bucle `for` (campo `in` objeto)?

Itera sobre las claves (nombres de las propiedades) del objeto, no sobre sus valores.

c. ¿Qué diferencia hay entre acceder a una propiedad con `usuario.nombre` y `usuario['nombre']`?

La notación de punto es más directa pero requiere saber el nombre exacto. La notación de corchetes permite usar variables dinámicas o nombres de propiedades con caracteres especiales.

5. Funciones en JavaScript

a. ¿Qué es una función anónima?

Es una función que no tiene nombre, utilizada comúnmente como callback o asignada a variables.

b. ¿Qué es una IIFE? Pon un ejemplo.

Es una función que se define y se ejecuta inmediatamente.

Ejemplo: `(function() { console.log('Hola'); })();`

c. ¿Qué ventajas ofrecen las funciones flecha?

Tienen una sintaxis más corta y no crean su propio contexto de `this` (heredan el del padre), lo que evita errores comunes en objetos y clases.

6. Temporizadores

a. ¿Qué diferencia hay entre `setTimeout()` y `setInterval()`?

`setTimeout()` ejecuta la función una sola vez tras el tiempo indicado. `setInterval()` la ejecuta repetidamente en bucle cada cierto tiempo.

b. ¿Qué hace `clearTimeout()`? ¿Y `clearInterval()`?

Sirven para detener la ejecución de los temporizadores `setTimeout` y `setInterval` respectivamente, usando el ID que devolvieron al crearse.

7. Arrays y colecciones

a. ¿Para qué sirve `.filter()` en un array?

Crea un nuevo array que contiene únicamente los elementos que cumplen una condición

b. ¿Qué diferencia hay entre Map y Array?

Un Array es una lista ordenada por índices numéricos mientras que un Map es una colección de pares clave-valor donde la clave puede ser de cualquier tipo (objetos, funciones, etc.) y no solo strings o números.