Desarrollo Web en Entornos de Cliente

Prueba práctica - JavaScript fundamentals

En las últimas unidades didácticas hemos estado viendo los fundamentos de JavaScript. En esta prueba práctica, aplicarás esos conocimientos para resolver los siguientes ejercicios. Recuerda que es fundamental comentar tu código y seguir las buenas prácticas de programación. Asegúrate de validar las entradas de los usuarios en los ejercicios que las requieran.

Ten en cuenta que SOLO está permitido el uso de internet para la consulta de la siguiente documentación:

- Free Code Camp
- JavaScript Info
- MDN Web Docs
- LenguajeJS
- regexr.com

No está permitido el uso de ninguna herramienta de IA.

Ejercicio 1: Conversor de temperaturas (criterios a evaluar: 2b, 2d, 2h)

Crea un script que permita convertir entre diferentes escalas de temperatura (Celsius, Fahrenheit y Kelvin) utilizando las fórmulas de conversión.

1. **Entradas:**
   - Pedir al usuario que ingrese una temperatura en Celsius.
   - Pedir al usuario a qué escala desea convertir (Fahrenheit, Kelvin).
2. **Proceso:**
   - Validar los datos introducidos por el usuario para evitar errores en el script o un mal funcionamiento del mismo.
   - Realizar la conversión según las siguientes fórmulas:
     - Fahrenheit = (Celsius * 9/5) + 32
     - Kelvin = Celsius + 273.15
3. **Salida:**
   - Mostrar el resultado al usuario (por consola).
   - Incluir manejo de errores para entradas inválidas y mostrar mensajes de error con alert().

Ejercicio 2: Entrega de carnet (criterios a evaluar: 2c, 2e, 4b)

Implementa una función llamada `darCarnet()` que reciba un número (entre 16 y 120), y un booleano que determinará si está estudiando o no. Devolverá el carné correspondiente.

1. **Entrada**
   - La función recibirá:
     - Un número entre 16 y 120.
     - Un booleano que indique si está estudiando.
2. **Proceso**:
   - Crear una función darCarnet() que hará lo siguiente:
     - Validar los datos recibidos para evitar errores en el script o un mal funcionamiento del mismo.
     - Utilizar estructuras de decisión para devolver el carnet a entregar:
       - 16-25 : carnet joven.
       - 26-65 : sin carnet
       - 66-120 : carnet de la tercera edad
       - Estudiante : carnet de estudiante
         (*) La entrega del carnet de estudiante prevalece sobre las consideraciones de edad.
   - Probar el funcionamiento de la función llamándola varias veces con distintos argumentos.
3. **Salidas**:
   - Mostrar un mensaje al usuario con el carnet correspondiente (por consola).
   - Incluir manejo de errores para entradas inválidas, debemos usar alert() para mostrar los mensajes de error).

Ejercicio 3: Gestión de lista de la compra (criterios a evaluar: 2f, 4c, 4e)

Desarrolla una función `agregarOEliminarElemento()` que gestione una lista de la compra.

1. **Entradas**:
    - La función recibe:
        - La lista
        - Un elemento a comprar.
2. **Proceso**:
    - Crear una función `agregarOEliminarElemento()` que acepte un array y un nombre de elemento y haga lo siguiente:
        - Validar los datos introducidos por el usuario para evitar errores en el script o un mal funcionamiento del mismo.
        - Si no existe el elemento en el array, lo añade al final.
        - Si existe el elemento en el array, lo elimina.
        - La existencia del elemento se debe comprobar buscándolo mediante estructuras repetitivas, no mediante métodos predefinidos.
        - Mostrar la lista actualizada después de cada agregación o eliminación.
    - Probar el funcionamiento de la función con un array y algunas llamadas.
3. **Salidas**:
    - Mostrar por consola todos los elementos en la lista al final de cada operación.
    - Incluir manejo de errores para entradas inválidas y mostrar mensajes de error con alert().

Ejemplo de Interacción con el Usuario

- La función es llamada con "Leche", y el programa la añade a la lista.
- La función es llamada con "Leche" de nuevo, y el programa elimina "Leche" de la lista.
- La función es llamada con "Pan", y el programa lo añade a la lista.
- Mostrar lista: Al final de cada operación anterior se muestra la lista actualizada de compras.

Ejercicio 4: Suma de ventas (criterios a evaluar: 2g, 4a, 4d)

Desarrolla una función `sumarArrays()` que reciba 2 arrays de ventas diarias realizadas por dos vendedores distintos cuyos elementos son cantidades en euros, como por ejemplo 4 €.
Si un array tiene menos elementos que otro, deberás igualarlos añadiéndole elementos de 0 € por el final con algún método predefinido de array.
Devuelve un tercer array, donde cada elemento del índice x, será la suma de los elementos del índice x de los otros arrays (sin el €).

1. **Entradas**:
   - 2 arrays de cantidades en euros.
2. **Proceso**:
   - Validar los datos de los arrays.
   - Completar uno de los arrays si es necesario.
   - Crear y devolver el tercer array con los elementos sumados, usando funciones predefinidas que permitan ignorar el símbolo € para poder sumar.
3. **Salidas**:
   - Mostrar los 2 arrays completos, y debajo el resultante.
   - Incluir manejo de errores para entradas inválidas y mostrar mensajes de error con alert().

Ejemplo de array recibidos:

Array de entrada 1	500 €	300.5 €	120 €	600 €
Array de entrada 2	200.75 €	350 €		

Los arrays del ejemplo tras ejecutar el proceso:

Array de entrada 1 completo	500 €	300.5 €	120 €	600 €
Array de entrada 2 completo	200.75 €	350 €	0 €	0 €
Array de salida	700.75	650.5	120	600

Ejercicio 5: Clase Personaje para rol (criterios a evaluar: 4f, 4g, 4h, 4i, 4k)

Crea una clase `Personaje` que represente personajes en un juego de rol en web. La clase debe tener las siguientes propiedades y métodos:

1. **Propiedades**:
   - Nombre (string)
   - Tipo (string: “Enano”, “Elfo” o “Humano”)
   - Experiencia (número, mayor o igual que 0)
   - Vidas (número, mayor o igual a 0)
2. **Métodos**:
   - `constructor()`: Inicializa el objeto con el nombre, tipo y vidas indicados por argumentos. La experiencia será 0. Se deben validar los argumentos.

- `enfrentarMonstruo()`: Recibe por argumento el nivel del monstruo y calcula un entero aleatorio entre 0 y 100. Si el número aleatorio es inferior al nivel del monstruo, el personaje perderá una vida. Si es igual o superior, sumará a su experiencia el nivel del monstruo. Personajes con 0 vidas al iniciar el enfrentamiento no pueden enfrentarse a monstruos, por lo que si tiene 0 vidas no sucede nada. El nivel del monstruo debe estar entre 0 y 100. Valida el nivel y el número de vidas del personaje.
- `mostrarInfo()`: Devuelve un string con la información del personaje (nombre, tipo, experiencia y vidas).

3. **Instancia**:
   - Crea un objeto de la clase `Personaje` y enfréntalo a algunos monstruos, mostrando por consola cómo queda tras cada combate.
4. **Salidas**:
   - Incluir manejo de errores para entradas inválidas y mostrar mensajes de error con alert().

Entrega

La entrega se realizará a través de la plataforma educativa moodle centros en la entrega habilitada para tal efecto. Los ejercicios deben entregarse siguiendo la siguiente estructura y nombres de archivo:

- Ejercicio 1:
  - e1-Apellido1Apellido2-Nombre.js
  - e1-Apellido1Apellido2-Nombre.html

- Ejercicio 2:
  - e2-Apellido1Apellido2-Nombre.js
  - e2-Apellido1Apellido2-Nombre.html

- Ejercicio 3:
  - e3-Apellido1Apellido2-Nombre.js
  - e3-Apellido1Apellido2-Nombre.html

- Ejercicio 4:
  - e4-Apellido1Apellido2-Nombre.js
  - e4-Apellido1Apellido2-Nombre.html

- Ejercicio 5:
  - e5-Apellido1Apellido2-Nombre.js
  - e5-Apellido1Apellido2-Nombre.html

Importante:

- Si los ejercicios no se ejecutan, no se procederá a la corrección de los mismos.
- Si los ejercicios no siguen la estructura y/o nombres indicados, no se procederá a la corrección de los mismos.
- Si se detecta plagio o copia entre compañeros/-as, la calificación será de 0 puntos en todas las partes de las que se compone el trabajo para ambos/-as.
- Igualmente, si se detecta plagio o copia de internet, la calificación será de 0 puntos en todas las partes de la que se compone el trabajo.
- Si alguna de las partes es entregada fuera de plazo, no se procederá a la corrección de la misma.

Resultados de Aprendizaje y Criterios a Evaluar

Resultados de Aprendizaje (RAs):

2. Escribe sentencias simples, aplicando la sintaxis del lenguaje y verificando su ejecución sobre navegadores web.
- b) Se han utilizado los distintos tipos de variables y operadores disponibles en el lenguaje.
- c) Se han identificado los ámbitos de utilización de las variables.
- d) Se han reconocido y comprobado las peculiaridades del lenguaje respecto a las conversiones entre distintos tipos de datos.
- e) Se han utilizado mecanismos de decisión en la creación de bloques de sentencias.
- f) Se han utilizado bucles y se ha verificado su funcionamiento.
- g) Se han añadido comentarios al código.
- h) Se han utilizado herramientas y entornos para facilitar la programación, prueba y depuración del código.

4. Programa código para clientes web analizando y utilizando estructuras definidas por el usuario.
- a) Se han clasificado y utilizado las funciones predefinidas del lenguaje.
- b) Se han creado y utilizado funciones definidas por el usuario.
- c) Se han reconocido las características del lenguaje relativas a la creación y uso de matrices (arrays).
- d) Se han creado y utilizado matrices (arrays).
- e) Se han utilizado operaciones agregadas para el manejo de información almacenada en colecciones.
- f) Se han reconocido las características de orientación a objetos del lenguaje.
- g) Se ha creado código para definir la estructura de objetos.
- h) Se han creado métodos y propiedades.
- i) Se ha creado código que haga uso de objetos definidos por el usuario.
- k) Se ha depurado y documentado el código.

Rúbricas de Evaluación

Rúbrica para Ejercicio 1: Conversión de temperaturas

Indicador	Excelente (2pt)	Aceptable (1pt)	Insatisfactorio (0pt)
Uso de Variables y Operadores (2pt)	Se ha utilizado adecuadamente variables y operadores, demostrando un sólido entendimiento (2pt)	Se ha utilizado variables y operadores de manera correcta, pero podría mejorar su uso (1pt)	Se han cometido errores significativos en el uso de variables y operadores (0pt)
Conversión entre Tipos de Datos (2pt)	Se ha reconocido y comprobado las conversiones entre tipos de datos de manera precisa (2pt)	Se ha identificado conversiones entre tipos de datos, pero la explicación es parcial o poco clara (1pt)	No se ha identificado adecuadamente las conversiones entre tipos de datos (0pt)
Validación de Entradas (2pt)	Se ha validado correctamente que los datos ingresados sean válidos y manejados adecuadamente (2pt)	Se ha intentado validar las entradas, pero la implementación es parcial (1pt)	No se ha realizado ninguna validación de entradas (0pt)
Comentarios en el Código (2pt)	Se han agregado comentarios de manera efectiva y coherente al código (2pt)	Se han utilizado comentarios, pero podrían ser más detallados o coherentes (1pt)	Se han utilizado pocos o ningún comentario en el código (0pt)
Calidad del Código (2pt)	El código está bien estructurado, es legible y sigue las mejores prácticas de programación (2pt)	El código es comprensible, pero podría beneficiarse de una mejor organización (1pt)	El código es difícil de seguir y carece de claridad (0pt)

Rúbrica para Ejercicio 2: Entrega de carnet

Indicador	Excelente (2pt)	Aceptable (1pt)	Insatisfactorio (0pt)
Identificación de Ámbitos (2pt)	Se han usado los modificadores de ámbito conforme a las necesidades ámbito del problema (2pt)	Se han usado solo algunos modificadores de ámbito de forma correcta. (1pt)	No se han usado, o se han usado de forma incorrecta los modificadores de ámbito de las variables(0pt)
Mecanismos de Decisión (2pt)	Se ha aplicado mecanismos de decisión de manera efectiva y lógica en el código (2pt)	Se ha utilizado mecanismos de decisión, pero podría mejorar su lógica (1pt)	No se ha aplicado adecuadamente mecanismos de decisión (0pt)
Uso de Funciones Definidas por el Usuario (2pt)	Se han creado y utilizado funciones definidas por el usuario de manera efectiva (2pt)	Se han creado funciones, pero su uso es ineficaz o poco claro (1pt)	No se han creado funciones definidas por el usuario (0pt)
Validación de Entradas (2pt)	Se han manejado adecuadamente las validaciones para evitar entradas incorrectas (2pt)	Se ha intentado validar, pero hay fallos en la implementación (1pt)	No se ha realizado ninguna validación de entradas (0pt)
Comentarios en el Código (2pt)	Se han incluido comentarios claros que explican la lógica del código (2pt)	Hay comentarios, pero no explican adecuadamente el funcionamiento (1pt)	No hay comentarios que expliquen el código (0pt)

Rúbrica para Ejercicio 3: Gestión de lista de compras

Indicador	Excelente (2pt)	Aceptable (1pt)	Insatisfactorio (0pt)
Uso de Bucles (2pt)	Se han utilizado bucles de manera efectiva para gestionar la lista (2pt)	Se ha utilizado un bucle, pero su implementación es ineficaz (1pt)	No se ha utilizado bucles (0pt)
Uso de Arrays (2pt)	Se han creado y utilizado arrays de manera efectiva para almacenar la lista de compras (2pt)	Se ha utilizado un array, pero su implementación es ineficaz (1pt)	No se ha creado ni utilizado un array (0pt)
Validación de Entradas (2pt)	Se ha validado correctamente que los elementos sean válidos antes de agregarlos o eliminarlos (2pt)	Se ha intentado validar las entradas, pero la implementación es parcial (1pt)	No se ha realizado ninguna validación de entradas (0pt)
Comentarios en el Código (2pt)	Se han agregado comentarios claros que explican la lógica del código (2pt)	Hay comentarios, pero no explican adecuadamente el funcionamiento (1pt)	No hay comentarios que expliquen el código (0pt)
Calidad del Código (2pt)	El código está bien estructurado, es legible y sigue las mejores prácticas de programación (2pt)	El código es comprensible, pero podría beneficiarse de una mejor organización (1pt)	El código es difícil de seguir y carece de claridad (0pt)

Rúbrica para Ejercicio 4: Suma de ventas

Indicador	Excelente (2pt)	Aceptable (1pt)	Insatisfactorio (0pt)
Uso de Funciones predefinidas (2pt)	Se han utilizado funciones predefinidas de manera efectiva (2pt)	Se han utilizado funciones predefinidas pero su uso es ineficaz o poco claro (1pt)	No se han usado funciones predefinidas (0pt)
Uso de Arrays (2pt)	Se han creado y utilizado arrays de manera efectiva para almacenar la lista de compras (2pt)	Se ha utilizado un array, pero su implementación es ineficaz (1pt)	No se ha creado ni utilizado un array (0pt)
Validación de Entradas (2pt)	Se ha validado correctamente que la entrada del usuario sea un número y esté en el rango permitido (2pt)	Se ha intentado validar las entradas, pero la implementación es parcial (1pt)	No se ha realizado ninguna validación de entradas (0pt)
Comentarios en el Código (2pt)	Se han agregado comentarios claros que explican la lógica del código (2pt)	Hay comentarios, pero no explican adecuadamente el funcionamiento (1pt)	No hay comentarios que expliquen el código (0pt)
Calidad del Código (2pt)	El código está bien estructurado, es legible y sigue las mejores prácticas de programación (2pt)	El código es comprensible, pero podría beneficiarse de una mejor organización (1pt)	El código es difícil de seguir y carece de claridad (0pt)

Rúbrica para Ejercicio 5: Clase Personaje para rol

Indicador	Excelente (2pt)	Aceptable (1pt)	Insatisfactorio (0pt)
Orientación a Objetos (2.5pt)	Se han reconocido las características de orientación a objetos del lenguaje de manera efectiva (2.5pt)	Se han utilizado características de orientación a objetos, pero la implementación es parcial (1.25pt)	No se han utilizado características de orientación a objetos (0pt)
Definición de Clases y Métodos (2.5pt)	Se ha creado código para definir la estructura de objetos, incluyendo métodos y propiedades (2.5pt)	Se ha creado código, pero la definición de métodos y propiedades es deficiente (1.25pt)	No se ha definido adecuadamente la estructura de objetos (0pt)
Depuración y Documentación (2.5pt)	Se ha depurado y documentado el código de manera efectiva (2.5pt)	Se ha realizado alguna depuración y documentación, pero es insuficiente (1.25pt)	No se ha depurado ni documentado el código (0pt)
Calidad del Código (2.5pt)	El código está bien estructurado, es legible y sigue las mejores prácticas de programación (2.5pt)	El código es comprensible, pero podría beneficiarse de una mejor organización (1.25pt)	El código es difícil de seguir y carece de claridad (0pt)