BOTIC-SOFOF-24-28-13-0081

Módulo 3: Fundamentos de Programación en JavaScript (60 Horas) / ELE069950089611

AE2_ABP-Ejercicio individual [Actividad Evaluada]

AE2_ABP-Ejercicio individual [Actividad Evaluada]

Ejercicio individual

Contexto

Imagina que estás desarrollando el sistema de puntuación de un minijuego en JavaScript. Este juego es muy simple: el jugador acumula puntos cada vez que realiza una acción, puede tener vidas extra según ciertas condiciones y recibe mensajes diferentes dependiendo del puntaje. Para esto, necesitas usar variables, operadores, estructuras condicionales y tipos de datos para modelar el comportamiento del juego.

Actividad

Desarrolla un programa en JavaScript que simule el sistema de puntuación de un juego aplicando los siguientes conceptos:

- **Crea variables** que representen el nombre del jugador, su puntuación inicial, número de vidas y nivel.
- Utiliza **nombres claros y significativos** para tus variables.
- Declara una **constante** que indique el puntaje necesario para ganar una vida extra.
- Usa **tipos de datos simples** (**string**, **number**, **boolean**) y **tipos complejos** (**arrays**, **objetos**) para organizar los datos del jugador.
- Implementa una función que simule una jugada:
  - Suma puntos al jugador.
  - Aumenta el nivel si el puntaje supera cierto umbral.
  - Muestra un mensaje con el estado actual del jugador usando concatenación de cadenas.

- Usa **operadores aritméticos**, de **incremento/decremento**, y **de comparación** para actualizar el estado del juego.
- Implementa la lógica con **if** y **else** para evaluar:
  - Si el jugador ha ganado una vida extra.
  - Si ha alcanzado el puntaje máximo.
  - Si debe finalizar el juego por quedarse sin vidas.

- Incluye al menos un **diagrama de flujo sencillo** (dibujado o explicado textualmente) que represente la lógica que implementaste.
- Asegúrate de manejar condiciones de borde, como puntaje negativo, exceso de vidas o valores inválidos.

Al finalizar, reflexiona brevemente sobre:

- ¿Cómo influyó el scope de las variables en el diseño de tu función?
- ¿Cuál fue el operador más útil y por qué?
- ¿En qué parte del código implementaste una validación para manejar un caso límite?

Entrega

- Archivo zip con tu código y además la reflexión.
- Ejecución: Individual.
- Duración: 1 jornada de clases.

Estado de la entrega

Estado de la entrega	Todavía no se han realizado envíos
Estado de la calificación	Sin calificar
Última modificación	-

DESARROLLO AE2_ABP-Ejercicio individual

1. Código juego.js

```
// Sistema de puntuación de un juego con objetos y múltiples jugadores

// Constantes
const puntaje_vida_extra = 100;
const maximo_vidas       = 5;
const puntaje_maximo     = 500;

// jugadores
let jugadores = [
   {
      nombre: "Juan",
      puntaje: 0,
      vidas: 3,
      nivel: 1,
      registro_partidas: []
   },
   {
      nombre: "Ana",
      puntaje: 0,
      vidas: 3,
      nivel: 1,
      registro_partidas: []
   }
];

// punto para tres rondas
let rondas = [
   [50, 120, 100],
   [80,  90, 110]
];

// inicio de ciclo
for (let i = 0; i < jugadores.length; i++) {
    let jugador = jugadores[i];
    console.log("Bienvenido " + jugador.nombre + ", empieza el juego!");
    console.log("-------------------------------------------------");

    for (let ronda = 0; ronda < rondas[i].length; ronda++) {
          let puntos_ganados = rondas[i][ronda];
          console.log("Ronda " + (ronda + 1) + " - Puntos ganados: " + puntos_ganados);

          if (puntos_ganados < 0) {
             console.log("Puntos negativos no permitidos. Jugada ignorada.");
             continue;
          }

          jugador.puntaje += puntos_ganados;
          jugador.registro_partidas.push(jugador.puntaje);
```

```
            if (jugador.puntaje >= 200 && jugador.nivel === 1) {
                jugador.nivel++;
                console.log("Nivel completado " + jugador.nombre + " acabas de subir al nivel 2");
            } else if (jugador.puntaje >= 400 && jugador.nivel === 2) {
                jugador.nivel++;
                console.log("Nivel completado " + jugador.nombre + " acabas de subir nivel 3");
            }

            // Vida extra si el puntaje es múltiplo de 100
            if (jugador.puntaje % puntaje_vida_extra === 0 && jugador.vidas < maximo_vidas) {
                jugador.vidas++;
                console.log("Bien hecho " + jugador.nombre + " ganaste una vida extra.");
            }

            if (jugador.vidas > maximo_vidas) {
                jugador.vidas = maximo_vidas;
            }

            if (jugador.vidas <= 0) {
                console.log("Game Over " + jugador.nombre + " te quedaste sin vidas.");
                break;
            }

            if (jugador.puntaje >= puntaje_maximo) {
                console.log("Felicitaciones " + jugador.nombre + " alcanzaste el puntaje máximo.");
            }

        // Mostrar estado del jugador
        console.log("Estado actual de " + jugador.nombre + ":");
        console.log("Puntaje: " + jugador.puntaje);
        console.log("Vidas  : " + jugador.vidas);
        console.log("Nivel  : " + jugador.nivel);
        console.log("-----------------------------------------------------");
    }
}
```

2. Desarrollo:

a. Diagrama de flujo textualmente

Inicio del juego por jugador

⬇

Obtener puntos ganados en la ronda

⬇

¿Los puntos son negativos?

Sí → Mostrar advertencia y saltar ronda

No → Continuar

⬇

Sumar puntos al total del jugador

⬇

Guardar en historial de puntajes

⬇

¿Puntaje >= 200 y está en nivel 1?
Sí → Subir a nivel 2

¿Puntaje >= 400 y está en nivel 2?
Sí → Subir a nivel 3

¿Puntaje divisible por 100?
Sí → Sumar una vida si no ha alcanzado el máximo

¿Puntaje >= 500?
Sí → Mostrar mensaje de puntaje máximo

¿Vidas <= 0?
Sí → Terminar juego para ese jugador

Mostrar estado actual del jugador

Repetir para la siguiente ronda

b. Validaciones de condiciones de borde

- ✓ Puntaje negativo: si el jugador gana puntos negativos, se muestra un mensaje y no se actualiza su estado:

```
if (puntosGanados < 0) {
    console.log ("Puntos negativos no permitidos. Jugada ignorada.");
    continue;
}
```

- ✓ Límite de vidas: no puede superar 5:

```
if (jugador.vidas > maximo_vidas) {
    jugador.vidas = maximo_vidas;
}
```

- ✓ Fin del juego si vidas son 0:

```
if (jugador.vidas <= 0) {
    console.log("Game Over " + jugador.nombre + " te quedaste sin vidas.");
    break;
}
```

- ✓ Límite de puntaje máximo (500 puntos):

```
if (jugador.puntaje >= puntaje_maximo) {
    console.log("Felicitaciones " + jugador.nombre + " alcanzaste el puntaje
    máximo.");
}
```

3. Reflexión final

- ✓ Cómo influyó el scope de las variables en el diseño

Al usar objetos dentro de un arreglo, fue más fácil tener toda la información de cada jugador junta y ordenada. Esto ayudó a que el código se viera más claro y no se mezclaran los datos entre jugadores. Como no usamos funciones, todo se hizo en el bloque principal y dentro del ciclo for, lo que hizo más simple manejar el estado de cada jugador por separado.

- ✓ Cuál fue el operador más útil y por qué
  El operador módulo (%) fue de los más útiles porque me permitió saber si el puntaje era múltiplo de 100. Gracias a eso, pude hacer que se dieran vidas extra en esos casos:

  if (jugador.puntaje % puntaje_vida_extra === 0 && ….

  También fue útil el operador >= para verificar si se debía subir de nivel o si se alcanzaba el puntaje máximo.

- ✓ En qué parte del código implementaste una validación para manejar un caso límite
  Agregué varias validaciones para manejar situaciones extremas, por ejemplo::

  - Ignorar jugadas con puntos negativos.
  - Evitar que las vidas pasen de 5.
  - Terminar el juego si las vidas bajan a 0.
  - Detectar si se alcanzó el puntaje máximo de 500 puntos.

  Estas validaciones ayudan a que el juego funcione bien y no aparezcan errores raros..