Programación

Undécima semana - Clases

Enero 2022

Cruz García, Iago

Anotaciones previas

Estos ejercicios son para familiarizarse con el lenguaje, la sintaxis y cómo resolverlos. Los primeros son sencillos y se va incrementando la dificultad. A continuación se presentan una serie de instrucciones que son necesarias para la resolución de los ejercicios:

- **alert(*parámetro*):** esta instrucción permite mostrar por pantalla un cartel con texto para mostrar la solución de algunos ejercicios.
- **console.log(*parámetro*):** esta instrucción permite mostrar en consola (F12 en el navegador) la solución de algunos ejercicios o trazar el código para comprobar que todo se ejecuta correctamente.
- **prompt(*texto, ejemplo*):** Muestra en pantalla un recuadro de **texto** y un cuadro para introducir texto con un **ejemplo**.
- Para poder ejecutar código JavaScript en Visual Studio Code debéis crear un fichero JavaScript (miScript.js) y un HTML básico (index.html por ejemplo) y dentro de la etiqueta <head> escribir los siguiente:
    - <script src="miScript.js"></script> comillas incluidas
- Ahora que sabemos encapsular creando funciones o métodos, se pueden hacer los ejercicios en el mismo fichero, simplemente comentando las llamadas a métodos que no necesiteis.

```
ejercicio_1()
//ejercicio_2()
//ejercicio_3()
```

Ejercicios

IMPORTANTE: A partir de ahora algunos ejercicios deben hacerse en múltiples ficheros .js, por lo que en vez de entregar todos en un mismo main.js, será necesario dividirlos en directorios. Se aconseja la estructura de "PrácticaX_ejercicio1" y dentro el index.html y los ficheros .js necesarios.

Para estos ficheros, lo mejor es agrupar aquellas funciones o métodos que realicen tareas similares (entradas.js o salidas.js por ejemplo). En caso de duda, no importa que un método quede aislado en un fichero.

El fichero que realice las llamadas a los métodos, que aune toda la funcionalidad, debe nombrarse como main.js y no debe tener más que un método que se llame igual y una llamada a este mismo.

Ejercicio 0

Vamos a crear una especie de "juego" desde 0, por lo que todos los ejercicios de esta semana estarán encaminados hacia ello. Podéis añadir, eliminar o modificar características del mismo a vuestro antojo, pero deben tener las mismas funcionalidades básicas del tema de Clases.

Crea los ficheros "index.html", "main.js" y "enemigo.js". En "enemigo.js" crea una clase del tipo Enemigo, sin atributos y sin métodos.

Pista: class…{}

Ejercicio 1

Añadir en la clase Enemigo los atributos de vida, ataque, nombre y nivel. Puedes probar a crear un objeto en el main para probar.

Pista: constructor(...){...}

Ejercicio 2

Añade en la clase Enemigo los método de obtención de atributos, getters, y los métodos de modificación de atributos, setters, para cada atributo (vida, ataque, nombre y nivel). Puedes probarlos creando un objeto en el main y comprobar que funcionan.

Pista: get y set.

Ejercicio 3

Crea un método en la clase Enemigo que permita reducir su vida dependiendo de la cantidad recibida por parámetro.

Pista: Recordemos que los métodos o funciones pueden recibir parámetros y devolverlos.

Ejercicio 4

Crea un método en la clase Enemigo que reciba por parámetro un valor y devuelva la experiencia que se obtiene al matarlo, que será su nivel multiplicado por 5 y dividido por el valor del parámetro:

Experiencia = (nivel x 5) / valor

Ejercicio 5

Crea un método en la clase Enemigo que compruebe si está muerto o sigue vivo. Para ello, comprueba la vida que le queda y si es 0 o inferior, devolverá *true*. Si no, *false*.

Prueba que funcionan los métodos y atributos creados hasta ahora con diferentes valores y muestra por pantalla los resultados para corroborar su uso.

Ejercicio 6

Vamos a generar nuestro héroe para jugar. Crea una fichero "heroe.js" y en el una clase de tipo Heroe.

Ejercicio 7

Añade a la clase de heroe los atributos de vida, ataque, nombre, nivel y experiencia.

Ejercicio 8

Añade a la clase de heroe los métodos getter y setter para cada atributo.

Ejercicio 9

Añade a la clase de heroe el mismo método de permite reducir vida que para la clase Enemigo. Haz lo mismo con el de comprobar si está muerto el héroe.

Ejercicio 10

Añade a la clase de heroe un método que reciba por parámetro un valor, sume ese valor a la experiencia y, si la experiencia es igual a 10 o superior, suma 1 nivel e iguala la experiencia a 0.

Ejercicio 11

Añade un método a la clase heroe que devuelva el valor de ataque multiplicado por el nivel.

Ejercicio 12

Ya tenemos nuestros dos personajes del juego. Vamos a hacer que hereden del mismo padre ya que comparten algunos métodos y atributos.

Crea el fichero "personaje.js" y en el mismo fichero una clase del tipo Personaje. Añádele los atributos que comparten ambos Enemigo y Heroe (vida, ataque, nombre y nivel), los getter y setter correspondientes a esos atributos, así como el método de recibir daño y el de comprobar si muere.

Ejercicio 13

Ahora que tenemos una clase padre, vamos a cambiar los hijos.

Haz que tanto la clase Enemigo como la clase Heroe, hereden de Personaje. Comenta los atributos y métodos que compartan con Personaje.

Pista: …extends… y super(...)

Ejercicio 14

Ahora el código debería ser más limpio y correcto. Vamos a probar que todo funciona haciendo una partida.

Crea un personaje (puedes pedir al usuario el valor de nombre y el resto introducirlos en código para evitar hacer trampas!) y un enemigo.

Crea un bucle while con una variable ultimo_turno que comience en 0. Esta variable nos indicará quién atacó el turno anterior. La condición de salida del bucle será si uno de los dos, personaje o enemigo, está muerto: while ('personaje'.'metodo_de_comprobacion' | | 'enemigo'.'metodo_de_compro bacion')

Siempre empezará el Heroe atacando y se alternará cada turno con el Enemigo. Para ello, comprueba si el valor de ultimo_turno es 0 o 1 para el Heroe y -1 para el Enemigo. Al finalizar el turno del Heroe, el valor de ultimo_turno pasará a ser -1. Si el que finaliza es el del enemigo, el valor de ultimo_turno será 1.

En cada turno, el Heroe golpeará al enemigo, utilizando los métodos que creamos. Cuando sea el turno del enemigo, lo mismo. Muestra por pantalla los valores de ambos personajes (nombre y vida para distinguirlos) para comprobar el transcurso de la batalla.

Así tendríamos nuestro pequeño juego terminado.

Ejercicio extra

Ya tenemos un juego bastante simple pero resultón. Hemos utilizado todas las capacidades esenciales que nos brindan las clases en JavaScript.

Os propongo una serie de apartados para completar el juego si os veis con ganas o para repasar:

1. La partida no termina solo al morir uno de los dos. Si el heroe gana, obtendrá los puntos de experiencia y podrá elegir si continuar la aventura, enfrentándose a más Enemigos.
2. Hay múltiples enemigos, pudiendo aparecer al azar o dependiendo del nivel del Heroe.
3. Se generan enemigos al azar dependiendo del nivel del Heroe. Los nombres pueden escogerse de un array, al azar.

El juego puede extenderse hasta el infinito, os animo a probar y divertiros.