Programación

Decimonovena semana

Marzo 2022

Cruz García, Iago

Anotaciones previas

Ejercicios

- Ejercicio 0
- Ejercicio 1
- Ejercicio 2
- Ejercicio 3
- Ejercicio 4
- Ejercicio 5
- Ejercicio 6
- Extra

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

Ahora que sabemos recuperar y almacenar datos en la sesión del navegador e incluso de forma semipermanente entre sesiones, vamos a mejorar algunas funciones de nuestro juego del héroe y el enemigo. Podéis utilizar cualquier versión con este fin, estos ejercicios tendrán en cuenta la implementación con eventos para mayor interactividad.

Dejo a vuestra elección qué tipo de almacenamiento utilizar (si SessionStorgae o LocalStorage), pues puede dar pie a múltiples interpretaciones o preferencias el cómo guardar los datos.

Añade al HTML dos radio button para modificar el tamaño de la fuente, uno que ponga "12px" y otra "26px". Crea el evento que modifique el tamaño de la fuente de todo el juego (acuerdate que establecer el tamaño del texto a toda la etiqueta body, modifica el tamaño de todo).

Guarda la selección en la sesión, de tal manera que al volver al HTML se mantenga el tamaño elegido.

Esto sería algo así:

index.html

```
<html id="html">

<head></head>

<body id="cuerpo">
        <input type="radio" name="tamTexto" class="tamTexto"
```

```
value="16px">16px
        <br>
        <input type="radio" name="tamTexto" class="tamTexto"
value="20px">20px
</body>
<footer>
        <script src="main.js"></script>
</footer>

</html>
```

main.js

```
function __main__() {
        preload();
        var radio = document.getElementsByClassName("tamTexto");
        for (var i = 0; i < radio.length; i++) {
        radio[i].addEventListener("click", (event) => {
        cambiaTexto(event);
        });
        }
}

function cambiaTexto(event) {
        var cuerpo = document.getElementById("cuerpo");
        cuerpo.style.fontSize = event.currentTarget.value;
        console.log(event.currentTarget.value);
        localStorage.setItem("tamFuente", event.currentTarget.value);
}

function preload() {
        var cuerpo = document.getElementById("cuerpo");
        var tamFuente = localStorage.getItem("tamFuente");
        cuerpo.style.fontSize = tamFuente;
        var radio = document.getElementsByClassName("tamTexto");
        for (var i = 0; i < radio.length; i++) {
        if (radio[i].value == tamFuente) {
        radio[i].checked = true;
```

```
        } else {
        radio[i].checked = false;
        }
        }
}

__main__();
```

Tomad esto como ejemplo e implementadlo en vuestro programa como veais conveniente. El resto de ejercicios seguirán una estructura similar, sobre todo para la carga de datos.

Ejercicio 1

Crea algunos radio button para modificar el color de la fuente del héroe. Guarda esos valores en el navegador para que mantenga esos valores al recargar la página.

Ejercicio 2

Crea un área de texto (textArea) para almacenar anotaciones del jugador. Guarda estas anotaciones. Puedes incluir un botón de guardar o simplemente guardar la información cada vez que el jugador interactúe con el textArea. Cuando se cargue la página de nuevo, esas anotaciones deben de estar presentes.

Ejercicio 3

Haz que el jugador pueda otorgarle un nombre al héroe. Cuándo se cargue la partida de nuevo, no preguntará por el nombre en primera instancia, si no que preguntará si quiere conservar dicho nombre.

Pista: Utiliza dos variables: "nombre" y "existeNombre" para hacer la comprobación.

Ejercicio 4

Crea algunos radio button para modificar el color del fondo, tanto del héroe como del enemigo. Guarda esos valores en el navegador para que mantenga esos valores al recargar la página.

Ejercicio 5

Crea un botón que permita eliminar todos los datos guardados en el navegador.

Ejercicio 6

Crea un grupo de checkbox que permita modificar las propiedades del texto para mostrarlo en **negrita** o *cursiva*. Al no ser propiedades excluyentes, deben poderse activar al mismo tiempo. Guarda los valores en el navegador para que se mantengan al recargar la página.

Extra

Prueba a guardar los datos de la partida al completo. Esto es decir, nombre del héroe, vida, experiencia, nivel, fase… Puedes hacerlo de múltiples formas, recomiendo hacer una clase "partida guardada" que guarde los valores y tenga el método "guardar partida" para almacenarla en el navegador y "cargar partida" para recogerlos.