Tabla de contenido

Ejercicio 1

Explica de forma concisa y clara el subsistema “Motor de renderizado” de la estructura de ejecución de un navegador. Especifica los utilizados actualmente por los navegadores principales.

Los motores de renderizado se encargan de transformar los archivos HTML, CSS y JavaScript en un formato adecuado para que el usuario lo pueda interpretar.
Este término se hizo popular gracias al proyecto Mozilla que fue el primero en separar el motor de renderizado del resto de navegador y así ser reutilizado.
Estos motores pueden interpretar el código de maneras diferentes, por eso es importante probar el código en varios navegadores.

Gecko: Desarrollado por Mozilla, es un software libre y de código abierto. El navegador más conocido que usa este motor es Firefox, SeaMonkey también lo utiliza aunque sea menos conocido.

WebKit: WebKit esta desarrollado por Apple para implementar su navegador Safari. También se utilizó para las primeras versiones de Chrome.

Blink: Desarrollado con contribuciones de Google, Opera, ASA, Intel y Samsung entre otras a partir de WebKit (fork) como parte del proyecto Chromium. Lo utiliza Chrome, Opera y Microsoft Edge (versiones recientes).

Ejercicio 2

Declara seis variables utilizando nombres acordes a su contenido:

- el precio de una sola rosa (8) y el número de rosas que tienes (70)
- el precio de un solo lirio (10) y el número de lirios que tienes (50)
- el precio de un solo tulipán (2) y la cantidad de tulipanes que tienes (120)

a) Ahora declara tres variables, una para cada una de las rosas, lirios y tulipanes que tienes, en las que colocas su precio total. Inserta los valores correspondientes en las variables utilizando las variables declaradas en el paso anterior.

b) Finalmente, declara una variable en la que almacenes el precio de todas tus flores (nuevamente, usa las variables anteriores para la inicialización). Muestra toda la información del inventario en la consola de la siguiente forma:

Rosa: precio unitario: 8 , cantidad: 70 , valor: 560
Lirio: precio unitario: 10 , cantidad: 50 , valor: 500
Tulipán: precio unitario: 2 , cantidad: 120 , valor: 240
Total: 1300

Resolución en el archivo “101SRS_1.html”. Link al archivo en GitHub.

Ejercicio 3

Uso de “*var*” en JavaScript

1. Define una variable saludo=”Hola” como variable local dentro de una función saludar() que muestre una ventana emergente con el valor de la variable saludo
   - Invoca la función
   - Muestra la variable desde fuera de la función mediante un window.alert

2. Define una variable global despedida=”Adiós” fuera de una función despedir( ) que muestre una ventana emergente con el valor de la variable despedida.
   - Invoca la función
   - Muestra la variable desde fuera de la función

3. Define dos variables con el mismo nombre y distinto valor: una fuera y otra dentro de una función ámbito() que muestre un window.alert de la misma. Muestra el valor de las dos mediante window.alert e invocando a la función.

4. Declara y define una variable global. Redefínela dentro de una función que la muestre mediante un window.alert. Muestra el valor de la variable mediante window.alert e invocando a la función.

5. Averigua qué son las variables automáticamente globales. Pon un ejemplo.

   Son variables que han sido declaradas sin utilizar ninguna palabra reservada (var, let o const). Estas variables se convierten automáticamente en globales.

```
variableAutomatica = "Pizza";

function varAuto() {
        variableAutomatica = "Queso";
        alert(variableAutomatica);
}

alert(variableAutomatica); // Imprime pizza
varAuto(); // Imprime queso
alert(variableAutomatica); // La variable se ha sobreescrito al ser global
```

Resolución en el archivo “101SRS_2.html”. Link al archivo en GitHub.

Conclusiones

Con esta práctica aprendemos:

- Los motores de renderizado son una parte muy importante en el desarrollo web ya que depende de ellos que las web se visualicen correctamente.

- También a entender el scope o ámbito de las variables en JavaScript. Es muy importante **no usar** la palabra reservada “var” ya que al ser de ámbito global puede traernos bastantes problemas a lo largo del desarrollo de la aplicación. Aunque tengamos bloques de varios <scripts>, usando “var” la variable seguirá siendo visible para el resto.

Bibliografía y Fuentes consultadas

Motores de renderizado

- Wikipedia. Motor de renderizado
- Blog Ctrly. Motores de render
- Platzi. Motores de renderizado
- Apuntes propios.

Variables automáticamente globales

- Scope en JavaScript