- ¿Qué diferencia existe entre == y === en JavaScript?

== realiza una comparación de igualdad sin considerar el tipo de dato, mientras que === si lo hace.

- ¿Qué describe mejor el término "Scope" en programación?

La duración de vida y accesibilidad de una variable.

- ¿Cuál es la diferencia principal entre let y var en JavaScript?

let es para declarar variables locales y var para variables globales.

- ¿Qué hace el algoritmo de ordenamiento "Bubble Sort"?

Organiza un array comparando elementos adyacentes y cambiándolos si están en el orden incorrecto.

- ¿Cuál es el propósito del operador ternario (condición ? expr1 : expr2) en JavaScript?

Simplificar la escritura de condicionales if/else en una única línea.

- ¿Cuál es la diferencia entre el operador && y el operador || en JavaScript?

&& realiza una comparación lógica "y" (AND), mientras que || realiza una comparación lógica "o" (OR).

- ¿Para qué se utilizan los objetos en programación?

Para crear estructuras de datos complejas y organizar información relacionada en propiedades y métodos.

- ¿Qué hace el ‘this’ en el ejemplo dado?

```
const persona = {
    nombre: "Lautaro",
    edad: 25,
    saludar: function() {
      console.log(`Hola, soy ${this.nombre} y tengo ${this.edad} años.`);
    }
  };

  persona.saludar();
```

Hace referencia al objeto persona.

- En el contexto de las "Funciones", ¿cuál es la principal diferencia entre una función declarada y una función expresada en JavaScript?

Una función declarada se define con la palabra clave ‘function’ y puede ser llamada antes de la declaración, mientras que una función expresada se asigna a una variable y solo puede ser llamada después de la asignación.

- ¿Cuál es el propósito principal de los "Diagramas de Flujo" en programación?

Representar gráficamente el flujo de control y la lógica de un programa mediante símbolos y conexiones.

- En el contexto de "Switch", ¿por qué es común utilizar la palabra clave "break" después de cada caso?

Para garantizar que solo se ejecuta el bloque de código correspondiente al caso coincidente y salir del bloque switch.

- ¿Cómo se podría implementar un algoritmo para calcular la suma de los elementos en la diagonal principal de esta matriz?

```
const matrizCuadrada = [
  [3, 1, 4],
  [1, 5, 9],
  [2, 6, 5]
];
```

```
function sumaDiagonalPrincipal(matriz) {
    let suma = 0;
    for (let i = 0; i < matriz.length; i++) {
      suma += matriz[i][i];
    }
    return suma;
  }
```