

Juan Neftaly Castellanos Hernández. 00182222
Investigar y practicar los siguientes conceptos

Objetos JSON

Los Objetos JSON (JavaScript Object Notation) son una forma de presentar los datos de manera clara y ordenada, para que tanto humanos como maquinas puedan entenderlo. Utilizan la sintaxis de clave - valor.

```
json Copy code  
  
{  
  "nombre": "Juan",  
  "edad": 30,  
  "ciudad": "México"  
}
```

Estructuras de Control

- Condicionales (if, else if, else): Las declaraciones condicionales permiten ejecutar cierto bloque de código si se cumple una condición.

```
if (condición) {  
  // Código a ejecutar si la condición es verdadera  
} else if (otraCondición) {  
  // Código a ejecutar si otraCondición es verdadera  
} else {  
  // Código a ejecutar si ninguna de las condiciones a  
}
```

- Bucles (for, while, do...while): Los bucles permiten repetir acciones mientras se cumple una condición o hasta que una condición se vuelva falsa.

For: Itera una fracción de código un número de veces.

```
javascript Copy code  
  
for (let i = 0; i < 5; i++) {  
  // Código a ejecutar en cada iteración  
}
```

While: ejecuta código hasta que se cumpla una condición.

```
javascript Copy code

while (condición) {
    // Código a ejecutar mientras la condición sea verdadera
}
```

Do While: Se ejecuta hasta que la condición se cumpla y hace algo hasta que esa condición termine.

```
javascript Copy code

do {
    // Código a ejecutar al menos una vez
} while (condición);
```

- Switch: Permite tomar decisiones basadas en el valor de una expresión. Se compara la expresión con varios casos y se ejecuta el bloque de código correspondiente al primer caso que coincida (o el bloque de código en la sección default si no hay coincidencias).

```
javascript Copy code

switch (expresión) {
    case valor1:
        // Código a ejecutar si expresión es igual a valor1
        break;
    case valor2:
        // Código a ejecutar si expresión es igual a valor2
        break;
    default:
        // Código a ejecutar si no se cumple ningún caso
}
```

Let - Const vs. Var

- Utiliza let cuando necesites variables que puedan cambiar de valor y tengan un alcance de bloque.
- Utiliza const cuando desees declarar constantes que no cambiarán de valor y tengan un alcance de bloque.
- Evita var en favor de let y const, ya que var puede causar problemas de alcance y es menos predecible en comparación con las otras dos.

Funciones (normales, anónimas y funciones flecha)

1. Funciones Normales: Las funciones normales se definen utilizando la palabra clave function. Tienen un nombre y pueden aceptar parámetros. Son útiles cuando necesitas utilizar un bloque de código en diferentes partes de tu programa.

```
javascript Copy code

function suma(a, b) {
    return a + b;
}

// Llamando a la función
const resultado = suma(2, 3); // resultado es igual a 5
```

2. Funciones Anónimas: Las funciones anónimas son funciones sin nombre y se definen generalmente dentro de expresiones o como argumentos para otras funciones. Son útiles en situaciones donde solo necesitas una función temporalmente o cuando deseas encapsular código.

```
javascript Copy code

const multiplicar = function(a, b) {
    return a * b;
};

// Llamando a la función anónima
const resultado = multiplicar(4, 5); // resultado es igual a 20
```

3. Funciones Flecha (Arrow Functions): Las funciones flecha son una forma más concisa de escribir funciones en JavaScript. Se definen con la sintaxis `() => {}` y no tienen su propio contexto `this`, lo que las hace útiles en ciertos contextos, como funciones de devolución de llamada (callbacks) o cuando se necesita mantener el valor de `this` de un contexto externo.

```
javascript Copy code

const resta = (a, b) => a - b;

// Llamando a la función flecha
const resultado = resta(7, 3); // resultado es igual a 4
```

Además, si una función flecha toma un solo parámetro, puedes omitir los paréntesis alrededor del parámetro:

```
javascript Copy code

const cuadrado = x => x * x;
```