Arreglos y Objetos 

Objetivos

- Conocer que son los arreglos
- Acceder a los elementos de un arreglo
- Recorrer un arreglo usando ciclos
- Interpolacion
- Agregar datos al dom desde un un ciclo interpolando datos de un array
- Objetos
- Insertar template con datos de un objeto
- Arreglos de objetos

¿Qué son los arreglos?

En JavaScript, los arreglos (también conocidos como arrays) son estructuras de datos que se utilizan para almacenar una colección ordenada de elementos. Estos elementos pueden ser de cualquier tipo de dato, como números, cadenas, objetos, funciones, otros arreglos, etc. Los arreglos permiten agrupar múltiples valores en una sola variable, lo que facilita el manejo y la manipulación de datos.

Los arreglos en JavaScript se definen utilizando corchetes **[ ]**, y los elementos se separan por comas.

Ejemplo de sintaxis:

```
const miArreglo = [elemento1, elemento2, elemento3, ...];
```

Donde **elemento1, elemento2, elemento3,** etc., representan los valores o elementos que deseas almacenar en el arreglo.

Accediendo a los elementos de un arreglo

Para acceder a los elementos de un arreglo, se utiliza el nombre del arreglo seguido de corchetes **[ ]** y dentro de estos se coloca el índice del elemento que deseas acceder. El índice de un arreglo es un número entero que representa la posición del elemento dentro del arreglo.

Los índices de los arreglos comienzan desde 0, lo que significa que el primer elemento tiene un índice de 0, el segundo tiene un índice de 1, y así sucesivamente.

Ejemplo de acceso por indice:

```
const miArreglo = [10, 20, 30, 40];
Acceder al primer elemento (10)
const primerElemento = miArreglo[0];
Acceder al segundo elemento (20)
const segundoElemento = miArreglo[1];
```

Ejercicio de ejemplo

Utiliza el siguiente arreglo y muestra por consola los nombres de Juan, Fabian y Jorge

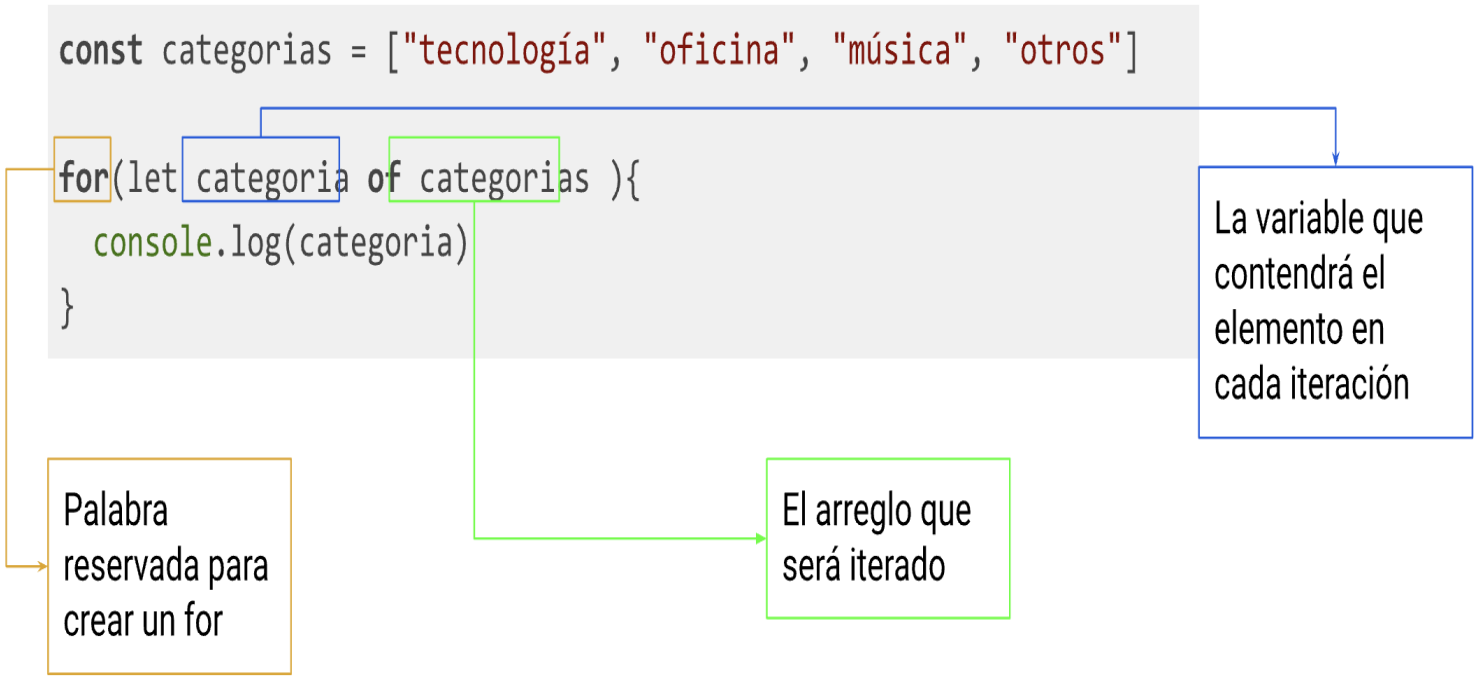
Ejemplo de acceso por indice:

```
const nombres = ["Juan", "Luisa", "Fabian", "Jorge"]
```

Ciclos

Los ciclos (también conocidos como bucles) en JavaScript son estructuras de control que te permiten ejecutar un bloque de código repetidamente hasta que se cumpla una condición específica. Los ciclos son esenciales para automatizar tareas repetitivas en la programación.

Anatomía de un ciclo



JavaScript existen varias formas de usar ciclos a continuacion veremos algunas:

Ciclo for:
El ciclo for es uno de los ciclos más comunes en JavaScript. Te permite ejecutar un bloque de código un número específico de veces.

sintaxis:

```
for (inicio; condición; incremento) {
  Código a ejecutar en cada iteración
}
```

Ejemplo recorriendo un arreglo:

```
const miArreglo = [10, 20, 30, 40];
for (let i = 0; i < miArreglo.length; i++) {
  Imprime cada elemento del arreglo
  console.log(miArreglo[i]);
}
```

Ciclo while:
El ciclo while se ejecuta mientras se cumpla una condición dada.

sintaxis:

```
while (condición) {
  Código a ejecutar en cada iteración
}
```

Ejemplo recorriendo un arreglo:

```
const miArreglo = [10, 20, 30, 40];
let indice = 0;

while (indice < miArreglo.length) {
  Imprime cada elemento del arreglo
  console.log(miArreglo[indice]);
  indice++;
}
```

Ciclo for...of:
Introducido en ECMAScript 2015 (ES6), el ciclo for...of se utiliza para recorrer elementos de objetos iterables, como arreglos, cadenas y más.

sintaxis:

```
for (const elemento of iterable) {
  Código a ejecutar en cada iteración
}
```

Ejemplo recorriendo un arreglo:

```
const miArreglo = [10, 20, 30, 40];

for (const elemento of miArreglo) {
  Imprime cada elemento del arreglo
  console.log(elemento);
}
```

Ciclo forEach:
método incorporado en JavaScript que se utiliza para iterar sobre elementos de un objeto iterable, como un arreglo. Proporciona una forma conveniente de ejecutar una función para cada elemento del arreglo sin tener que preocuparse por el manejo de índices o el control de la iteración.

Ejemplo forEach:

```
const miArreglo = [10, 20, 30, 40];

miArreglo.forEach((elemento) => {
  Imprime cada elemento del arreglo
  console.log(elemento);
});
```

El ciclo forEach es útil cuando solo necesitas recorrer los elementos del arreglo y realizar una operación en cada uno. Sin embargo, no se puede detener prematuramente como se podría hacer con break en un ciclo for.

Ejercicio de ejemplo

Mostrar por consola todos los elementos del arreglo utilizando un ciclo

Ejemplo forEach:

```
const lenguajes = ["JavaScript", "Java", "Python", "C++", "C#"];
```

Interpolación

Se refiere al proceso de combinar o insertar valores en una cadena de texto o una plantilla, para crear una cadena final que contiene los valores deseados. Es una forma conveniente de construir cadenas de texto más complejas que incluyen datos variables sin tener que concatenar manualmente las partes.

En el contexto de la programación, la interpolación es especialmente útil cuando deseas combinar variables u otros valores dentro de una cadena de texto. En lugar de usar la concatenación tradicional con el operador +, la interpolación proporciona una sintaxis más legible y menos propensa a errores.

En JavaScript, la interpolación se puede lograr utilizando plantillas de cadena (template strings o template literals) introducidas en ECMAScript 2015 (ES6) utilizando el delimitador de comillas invertidas **(``)**.

Ejemplo interpolación:

```
const nombre = "Juan";
const edad = 25;

const mensaje = `Hola, mi nombre es ${nombre} y tengo ${edad} años.`;
Imprimirá: Hola, mi nombre es Juan y tengo 25 años.
console.log(mensaje);
```

Concatenación

La concatenación es el proceso de unir dos o más cadenas de texto en una sola cadena. En JavaScript, la concatenación se puede lograr utilizando el operador +.

Ejemplo interpolación:

```
const nombre = "Juan";
const apellido = "Pérez";

const nombreCompleto = nombre + " " + apellido;
Imprimirá "Juan Pérez"
console.log(nombreCompleto);
```

Manipulando el DOM con arreglos, ciclos e interpolando

A continuacion veremos como podemos insertar datos en el dom mediante ciclos e interpolando los datos que estan en el arreglo esto nos data la ventaja de no tener que escribir codigo html para cada elemento que queremos agregar al dom y lo hace mas dinamico

Paso 1 vamos a definir el bloque en el cual vamos a insertar nuestros datos:

HTML:

```
<ul id="dynamic-content"></ul>
```

Paso 2 vamos a definir un array de nombres el cual vamos a recorrer con un for of y con la funcion innerHTML vamos a pintar los datos en en nuestra pagina web

JS:

```
const nombres = ["Juan", "Luisa", "Fabian", "Jorge"];
const dynamicContent = document.getElementById("dynamic-content");
for (const nombre of nombres) {
  dynamicContent.innerHTML += `<li>${nombre}</li>`;
}
```

Ejercicio de ejemplo

Agregar las notas musicales del siguiente arreglo al dom utilizando un ciclo for of

JS:

```
const escalaMayorDeDo = ["Do", "Re", "Mi", "Fa", "Sol", "La", "Si"];
```

¡Cuidado con innerHTML!

La propiedad innerHTML es una forma conveniente de insertar contenido en el DOM, pero tiene un inconveniente importante: cada vez que se llama a innerHTML, el navegador debe volver a analizar y renderizar el contenido de la página. Esto puede ser muy costoso en términos de rendimiento, especialmente cuando se inserta contenido repetidamente.

En su lugar, podemos usar la siguiente técnica para insetar el contenido.

Ej:

```
const data = ["Juan", "Luisa", "Fabian", "Jorge"];
let html = "";
const dynamicContent = document.getElementById("dynamic-content");
for (const nombre of data) {
  html += `<li>${nombre}</li>`;
}
dynamicContent.innerHTML = html;
```

Objetos

En JavaScript, los objetos son estructuras de datos que permiten almacenar y organizar colecciones de valores, funciones y otras propiedades. Los objetos son una parte fundamental de la programación en JavaScript y se utilizan para representar entidades o conceptos del mundo real, como usuarios, productos, pedidos y más.

Un objeto en JavaScript se compone de pares clave-valor, donde cada clave es una cadena (también conocida como propiedad) que se utiliza para acceder a su respectivo valor. Los valores pueden ser de cualquier tipo de dato, incluyendo números, cadenas, arreglos, funciones e incluso otros objetos. Las propiedades de un objeto pueden ser accedidas y manipuladas utilizando la notación de punto (objeto.propiedad) o la notación de corchetes (objeto['propiedad']).

Ejemplo de como de ve un objeto:

```
const persona = {
  nombre: 'Juan',
  edad: 30,
  esEstudiante: false
};
Imprimirá 'Juan'
Imprimirá 30
console.log(persona.nombre);
console.log(persona.edad);
```

Ejercicio

Dado el siguiente objeto mediante un console.log muestra el hobby

Ejemplo de como de ve un objeto:

```
const persona = {
  nombre: 'pedro',
  apellido: 'perez',
  profesion: 'front end developer',
  hobby: 'trekking',
};
```

Insertando datos de un objeto en un template

A continuacion veremos como podemos insertar datos de un objeto en un template para esto vamos a utilizar la interpolacion y la notacion de punto

HTML:

```
<section class="articulos"></section>
```

js:

```
const articulo = {
  id: 31,
  titulo: "Autos nuevos en chile",
  subtitulo: "el mercado de autos se normaliza",
  descripcion:
    "Lorem ipsum dolor sit amet consectetur adipisicing elit.",
};

const article = document.querySelector(".articulos");

  article.innerHTML = `
  <h2>${articulo.titulo}</h2>
  <h3>${articulo.subtitulo}</h3>
  <p>${articulo.descripcion}</p>
`;
```

Ejercicio

Crea un producto para un e-commerce a partir del siguiente objeto, no olvides crear el HTML y una sección para contener los datos que sé enviarán al procesar el objeto, usa interpolación y el innerHTML para insertar los datos.

js:

```
const producto = {
  id: 43,
  titulo: "cafetera magica",
  precio: 23990,
  color : "rojo",
  src: "https://picsum.photos/200/300",
  descripcion: "Lorem ipsum dolor sit amet consectetur adipisicing elit.",
}
```

Arreglos de Objetos

Un arreglo de objetos en JavaScript es una estructura de datos que almacena una colección de objetos. Cada elemento en el arreglo es un objeto en sí mismo, y estos objetos pueden contener propiedades y métodos que representan diferentes atributos y comportamientos.

Un arreglo de objetos es útil cuando deseas manejar múltiples entidades con características similares, pero cada entidad tiene sus propias propiedades y valores únicos. Por ejemplo, si estás trabajando con una lista de usuarios, productos, tareas o cualquier entidad con datos relacionados, puedes almacenar cada entidad como un objeto en un arreglo.

Ejemplo de arreglo de objetos:

```
const personas = [
  { nombre: 'Juan', edad: 25 },
  { nombre: 'María', edad: 30 },
  { nombre: 'Carlos', edad: 28 }
];

Imprimirá 'Juan'
console.log(personas[0].nombre);
Imprimirá 30
console.log(personas[1].edad);
```

Insertando datos desde un arreglo de Objetivos en un template

A continuacion veremos como podemos insertar datos de un arreglo de objetos en un template para esto vamos a utilizar la interpolacion, la notacion de punto y un ciclo for of.

HTML:

```
<section class="products"></section>
```

js:

```
const products = [
  {id: 1, nombre: "item 1", precio: 12000},
  {id: 2, nombre: "item 2", precio: 14000}
]

let html = "";

const productList = document.querySelector(".products");

for (const product of products) {
  html += `
    <div id="${product.id}">
      <h2>${product.nombre}</h2>
      <h3>${product.precio}</h3>
    </div>
  `
}

productList.innerHTML = html;
```

Ahora que ya conocemos los conceptos de los arreglos, vamos a realizar algunos ejercicios, mucha suerte