1. Introducción JavaScript

JavaScript es un lenguaje de programación de alto nivel que se utiliza principalmente en el desarrollo web. Es un lenguaje interpretado que se ejecuta en el lado del cliente, lo que significa que se ejecuta en el navegador web del usuario. Sin embargo, también puede ser utilizado en el lado del servidor a través de Node.js.

JavaScript se utiliza para agregar interactividad y funcionalidad a las páginas web. Permite a los desarrolladores crear elementos dinámicos, como efectos visuales, animaciones, manipulación del contenido de la página en tiempo real y validación de formularios. Además, se utiliza para interactuar con servidores web y realizar solicitudes asíncronas (AJAX), lo que permite cargar y enviar datos en segundo plano sin tener que recargar toda la página.

Con el tiempo, JavaScript se ha vuelto aún más versátil y se utiliza también en el desarrollo de aplicaciones de servidor, aplicaciones móviles, juegos y aplicaciones de escritorio. Frameworks y bibliotecas populares, como React, Angular y Vue.js, están construidos en JavaScript y permiten a los desarrolladores crear aplicaciones web complejas y escalables.

JavaScript se integra en HTML a través de etiquetas de script. Hay dos formas principales de incluir código JavaScript en un documento HTML: incrustarlo directamente en la página o vincularlo desde un archivo externo.

Incrustar JavaScript en HTML:

Puedes incrustar código JavaScript directamente en la página HTML utilizando la etiqueta <script>.

```
<!DOCTYPE html>
<html>
<head>
  <title>Ejemplo de JavaScript en HTML</title>
</head>
<body>
  <h1>Mi Página</h1>

  <script>
```

```
    // Código JavaScript incrustado
    alert('¡Hola, mundo!');
  </script>
</body>
</html>
```

En este ejemplo, el código JavaScript se coloca dentro de la etiqueta <script>. Podés escribir tu código directamente entre las etiquetas <script> y </script>. En este caso, el código JavaScript mostrará un cuadro de diálogo de alerta con el mensaje "¡Hola, mundo!" cuando se cargue la página.

Vincular JavaScript desde un archivo externo:

También podés vincular un archivo JavaScript externo a tu documento HTML utilizando la etiqueta <script> con el atributo src.

```
<!DOCTYPE html>
<html>
<head>
  <title>Ejemplo de JavaScript en HTML</title>
  <script src="script.js"></script>
</head>
<body>
  <h1>Mi Página</h1>
</body>
</html>
```

En este caso, el archivo script.js contiene el código JavaScript que deseas ejecutar. Asegúrate de que el atributo src tenga la ruta correcta hacia el archivo JavaScript.

Al vincular un archivo JavaScript externo, el código contenido en ese archivo se ejecutará cuando se cargue la página. Podés escribir tu código en el archivo script.js y el navegador lo cargará y ejecutará automáticamente.

2. Variables y Constantes

En JavaScript, una variable es un contenedor para almacenar valores. Puedes utilizar variables para almacenar diferentes tipos de datos, como números, cadenas de texto, booleanos, objetos y más. Las variables en JavaScript se definen utilizando la palabra clave var, let o const.

Declaración de variables con var:

```
var nombre = "Juan";  // Variable con el nombre "nombre" y el valor "Juan"

var edad = 25;        // Variable con el nombre "edad" y el valor 25

var isAdulto = true;  // Variable con el nombre "isAdulto" y el valor true


// Las variables declaradas con "var" pueden ser reasignadas

nombre = "Pedro";     // Reasignar el valor de la variable "nombre"
```

Declaración de variables con let:

```
let ciudad = "Barcelona";  // Variable con el nombre "ciudad" y el valor "Barcelona"

let cantidad = 10;             // Variable con el nombre "cantidad" y el valor 10


// Las variables declaradas con "let" también pueden ser reasignadas

ciudad = "Madrid";            // Reasignar el valor de la variable                                    "ciudad"
```

Declaración de constantes con const:

```
const PI = 3.14;    // constante con el nombre "PI" y el valor 3.14

const URL = "https://www.ejemplo.com";  // constante con el nombre "URL"


// Las variables declaradas con "const" no pueden ser reasignadas
```

// PI = 3.14159; // Esto generaría un error, ya que PI es una constante y no puede cambiar su valor

Es importante tener en cuenta que las variables declaradas con var tienen un alcance de función o global, mientras que las variables declaradas con let y const tienen un alcance de bloque, lo que significa que solo están disponibles dentro del bloque en el que se declaran.

Ejemplo

```
var nombre = "Ana";
var edad = 30;

console.log(nombre);          // Imprime "Ana" en la consola
console.log("La edad es " + edad);   // Imprime "La edad es 30" en la consola

// Realizando operaciones con variables
var suma = 5 + 3;
console.log("El resultado de la suma es: " + suma);    // Imprime "El resultado de la suma es: 8"

// Concatenando variables y texto
var mensaje = "Hola, mi nombre es " + nombre + " y tengo " + edad + " años.";
console.log(mensaje);    // Imprime "Hola, mi nombre es Ana y tengo 30 años."
```

Estos son solo algunos ejemplos básicos de cómo se utilizan las variables en JavaScript. Las variables son una parte fundamental de la programación, ya que te permiten almacenar y manipular datos de manera dinámica en tu código.

3. Consola

La consola se utiliza principalmente para imprimir mensajes, depurar y probar el código JavaScript. Proporciona una forma interactiva de ejecutar instrucciones y ver los resultados en tiempo real.

Abrir la consola:

En Google Chrome y Firefox, puedes abrir la consola haciendo clic derecho en una página web y seleccionando "Inspeccionar" o "Inspeccionar elemento". Luego, ve a la pestaña "Consola".

En Safari, ve al menú "Desarrollo" y selecciona "Mostrar consola web".

También puedes abrir la consola utilizando el atajo de teclado Ctrl + Shift + J en Windows/Linux o Cmd + Option + J en macOS.

Ejecutar código en la consola:

Una vez que la consola esté abierta, puedes escribir y ejecutar código JavaScript directamente en ella. Simplemente escribe el código y presiona la tecla "Enter" para ejecutarlo. Por ejemplo, puedes escribir console.log("Hola, mundo"); y luego presionar "Enter" para imprimir "Hola, mundo" en la consola.

Imprimir mensajes en la consola:

Puedes utilizar la función console.log() para imprimir mensajes o valores en la consola. Por ejemplo:

```
console.log("Este es un mensaje de prueba");

var nombre = "Juan";

console.log("El nombre es: " + nombre);
```

Depurar y probar el código:

La consola es una herramienta útil para depurar y probar el código JavaScript. Puedes utilizarla para verificar el valor de las variables, realizar seguimiento de errores, probar funciones y experimentar con el código.

Además de console.log(), la consola proporciona otras funciones útiles, como console.error(), console.warn(), console.info(), entre otras, que te permiten imprimir diferentes tipos de mensajes para facilitar la depuración y el seguimiento de errores.

4. Tipos de datos y operadores

Tipos de datos básicos:

- Números: Representan valores numéricos, ya sea enteros o decimales. Por ejemplo: 3, 3.14, -5.
- Cadenas de texto: Representan una secuencia de caracteres. Deben estar encerradas entre comillas simples o dobles. Por ejemplo: 'Hola', "Mundo".
- Booleanos: Representan un valor lógico que puede ser true (verdadero) o false (falso).
- Undefined: Indica que una variable no ha sido asignada o que una propiedad no existe.

Operadores en JavaScript:

- Operadores aritméticos: Se utilizan para realizar operaciones matemáticas. Incluyen suma (+), resta (-), multiplicación (*), división (/), módulo (%) y operadores de incremento (++) y decremento (--).
- Operadores de asignación: Se utilizan para asignar valores a variables. El operador de asignación básico es el signo igual (=), pero también existen operadores de asignación compuestos, como +=, -=, *=, /=.
- Operadores de comparación: Se utilizan para comparar dos valores y devuelven un valor booleano (true o false). Incluyen igualdad (==), desigualdad (!=), igualdad estricta (===), desigualdad estricta (!==), mayor que (>), menor que (<), mayor o igual que (>=), menor o igual que (<=).
- Operadores lógicos: Se utilizan para combinar expresiones lógicas y evaluar su veracidad. Los principales operadores lógicos son AND (&&), OR (||) y NOT (!).
- Operadores de concatenación: El operador de concatenación (+) se utiliza para unir cadenas de texto.

5. Promp, Alert y COnfirm

Las funciones prompt, alert y confirm en JavaScript. Estas funciones se utilizan para interactuar con el usuario en forma de diálogos simples en el navegador.

prompt:

La función prompt muestra un cuadro de diálogo que le pide al usuario que ingrese un valor. El usuario puede escribir un valor y hacer clic en "Aceptar" o "Cancelar". La función prompt devuelve el valor ingresado por el usuario como una cadena de texto, o devuelve null si el usuario hizo clic en "Cancelar".

Ejemplo de uso del prompt:

```
var nombre = prompt("Por favor, ingresa tu nombre:");

console.log("¡Hola, " + nombre + "!");  // Imprime un saludo usando el nombre ingresado por el usuario
```

En este ejemplo, se muestra un cuadro de diálogo en el navegador que solicita al usuario que ingrese su nombre. El valor ingresado por el usuario se guarda en la variable nombre y luego se utiliza para imprimir un saludo personalizado en la consola.

alert:

La función alert muestra un cuadro de diálogo de alerta al usuario con un mensaje especificado. Este cuadro de diálogo solo tiene un botón de "Aceptar" y no devuelve ningún valor.

Ejemplo de uso del alert:

```
alert("¡Bienvenido a nuestro sitio web!");  // Muestra un mensaje de bienvenida en un cuadro de diálogo
```

En este ejemplo, se muestra un cuadro de diálogo de alerta con un mensaje de bienvenida cuando se carga la página.

confirm:

La función confirm muestra un cuadro de diálogo de confirmación al usuario con un mensaje y dos botones: "Aceptar" y "Cancelar". Devuelve un valor booleano: true si el usuario hizo clic en "Aceptar" y false si hizo clic en "Cancelar".

Ejemplo de uso del confirm:

```
var confirmacion = confirm("¿Estás seguro de que deseas eliminar este elemento?");

if (confirmacion) {

  console.log("El elemento ha sido eliminado.");    // Se ejecuta si el usuario hace clic en "Aceptar"
```

```
} else {
  console.log("La eliminación ha sido cancelada.");  // Se ejecuta si el usuario hace clic en "Cancelar"
}
```

En este ejemplo, se muestra un cuadro de diálogo de confirmación con un mensaje preguntando si el usuario está seguro de que desea eliminar un elemento. Dependiendo de la respuesta del usuario, se imprime un mensaje en la consola.

Recuerda que estas funciones son específicas del entorno del navegador y no están disponibles en otros contextos de JavaScript, como Node.js.

Espero que estos ejemplos te ayuden a comprender cómo se utilizan prompt, alert y confirm para interactuar con el usuario en JavaScript.

6. Estructuras de control

JavaScript ofrece varias estructuras de control que permiten controlar el flujo de ejecución del código.

6.1. Estructura if/else:

La estructura if/else permite tomar decisiones basadas en condiciones. Si se cumple una condición, se ejecuta un bloque de código específico. Si la condición no se cumple, se puede ejecutar un bloque alternativo utilizando else.

```
var edad = 18;


if (edad >= 18) {
  console.log("Eres mayor de edad");
} else {
  console.log("Eres menor de edad");
}
```

En este ejemplo, se verifica si la variable edad es mayor o igual a 18. Si es así, se imprime "Eres mayor de edad". De lo contrario, se imprime "Eres menor de edad".

6.2. Estructura switch:

La estructura switch se utiliza para seleccionar uno de varios bloques de código para ejecutar, según el valor de una expresión.

```
var diaSemana = 1;
var nombreDia;

switch (diaSemana) {
  case 1:
    nombreDia = "Lunes";
    break;
  case 2:
    nombreDia = "Martes";
    break;
  case 3:
    nombreDia = "Miércoles";
    break;
  // ...
  default:
    nombreDia = "Día desconocido";
    break;
}

console.log("Hoy es " + nombreDia);
```

En este ejemplo, se utiliza la variable diaSemana para seleccionar el nombre del día correspondiente utilizando la estructura switch. Dependiendo del valor de diaSemana, se asigna el nombre del día apropiado a la variable nombreDia.

6.3.Estructuras de bucle:

Bucle for: Se utiliza para repetir un bloque de código un número específico de veces.

```
for (var i = 0; i < 5; i++) {
  console.log(i);
}
```

En este ejemplo, el bucle for se ejecutará cinco veces, imprimiendo los valores del 0 al 4 en la consola.

Bucle while: Se utiliza para repetir un bloque de código mientras una condición se cumpla.

```
var contador = 0;


while (contador < 5) {
  console.log(contador);
  contador++;
}
```

En este ejemplo, el bucle while se ejecutará siempre que la variable contador sea menor que 5. Se imprimirá el valor de contador y se incrementará en 1 en cada iteración.

Estas son solo algunas de las estructuras de control más comunes en JavaScript. Además, JavaScript también ofrece estructuras como el bucle do-while, bucles anidados y estructuras de control más avanzadas, como try-catch para manejo de excepciones.

Las estructuras de control son esenciales para controlar el flujo de ejecución del programa y permiten realizar diferentes acciones según ciertas condiciones. Al combinar estas estructuras de control, puedes crear algoritmos y lógica compleja en tu código JavaScript.

7. Funciones

En JavaScript, las funciones son bloques de código reutilizables que se definen una vez y se pueden llamar en cualquier momento para ejecutar ese bloque de código. Las funciones permiten estructurar y organizar el código, facilitando su mantenimiento y reutilización.

7.1.Definición de funciones:

Para definir una función en JavaScript, se utiliza la palabra clave function, seguida del nombre de la función, paréntesis que pueden contener parámetros separados por comas (opcional) y luego el bloque de código entre llaves.

```
// Definición de una función sin parámetros
function saludar() {
  console.log("¡Hola, bienvenido!");
```

```
}


// Definición de una función con parámetros
function sumar(a, b) {
  var resultado = a + b;
  console.log("La suma es: " + resultado);
}
```

7.2. Llamada a funciones:

Para utilizar una función en JavaScript, se realiza una llamada o invocación de la función utilizando su nombre seguido de paréntesis. Si la función tiene parámetros, se pueden proporcionar los valores correspondientes dentro de los paréntesis.

```
// Llamada a una función sin parámetros
saludar();


// Llamada a una función con parámetros
sumar(5, 3);
```

En este ejemplo, la función saludar se llama sin parámetros y muestra un mensaje en la consola. La función sumar se llama con los parámetros 5 y 3, realiza la suma y muestra el resultado en la consola.

7.3. Retorno de valores:

Las funciones en JavaScript también pueden devolver un valor utilizando la palabra clave return. El valor devuelto puede ser capturado y utilizado en el lugar donde se llamó la función.

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


var resultado = multiplicar(4, 5);
console.log("El resultado de la multiplicación es: " + resultado);
```

En este ejemplo, la función multiplicar toma dos parámetros, realiza la multiplicación y devuelve el resultado utilizando return. Luego, el valor devuelto se asigna a la variable resultado y se muestra en la consola.

Las funciones en JavaScript pueden tener cualquier cantidad de parámetros, pueden ser anónimas (sin nombre), asignadas a variables y pueden incluso ser pasadas como argumentos a otras funciones. Además, JavaScript también admite funciones flecha (arrow functions) introducidas en versiones más recientes.

Las funciones son una parte fundamental de JavaScript y son esenciales para la estructura y organización del código, así como para la reutilización de bloques de código.