JavaScript

¿Qué es la programación?

La programación es el proceso de crear software o aplicaciones informáticas mediante la escritura de código en un lenguaje de programación. El objetivo de la programación es crear instrucciones que una computadora pueda entender y ejecutar para realizar una tarea específica.

En general, la programación implica el diseño, desarrollo, prueba y mantenimiento de software. Los programadores utilizan una variedad de lenguajes de programación, herramientas y plataformas para desarrollar aplicaciones, desde pequeñas aplicaciones para dispositivos móviles hasta sistemas operativos de gran escala.

La programación también se considera una habilidad esencial en el mundo moderno, y cada vez más personas están aprendiendo a programar para aprovechar las oportunidades en el mercado laboral y para desarrollar sus propias ideas y proyectos.

¿Qué es la JavaScript?

Desde su creación en 1995, JavaScript ha evolucionado y crecido de manera constante hasta convertirse en uno de los lenguajes de programación más utilizados y populares en la actualidad.

En la década de 2000, la aparición de Ajax (Asynchronous JavaScript and XML) impulsó aún más la popularidad de JavaScript al permitir que las aplicaciones web se actualizaran de manera dinámica sin necesidad de cargar una página nueva completa. Esto hizo posible el desarrollo de aplicaciones web interactivas y de alta calidad, como Google Maps y Gmail.

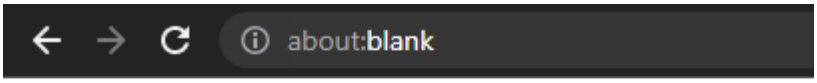
Con el creciente uso de JavaScript en el lado del cliente, se creó la necesidad de usar el lenguaje en el lado del servidor también. Esto llevó al desarrollo de Node.js, un entorno de ejecución de JavaScript del lado del servidor, en 2009, lo que permitió a los desarrolladores usar JavaScript para crear aplicaciones de servidor.

En los últimos años, JavaScript ha experimentado un auge en popularidad y ha continuado evolucionando y mejorando. Las últimas versiones de ECMAScript han introducido características avanzadas y herramientas para hacer que la escritura de código en JavaScript sea más fácil y eficiente, como los módulos, las promesas y el operador de propagación.

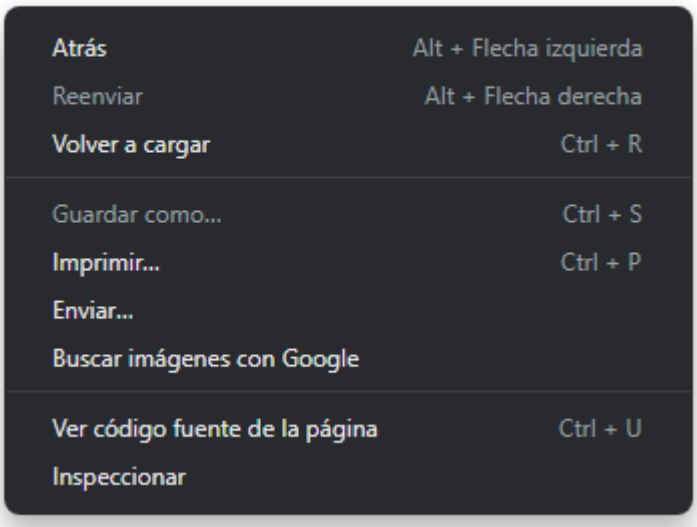
Hoy en día, JavaScript se utiliza en una amplia gama de aplicaciones, desde el desarrollo de aplicaciones móviles y de escritorio hasta la creación de aplicaciones web de última generación con tecnologías como React, Vue y Angular. Además, se ha convertido en uno de los lenguajes esenciales para la programación web y el desarrollo de la mayoría de los sitios web modernos.

Iniciemos

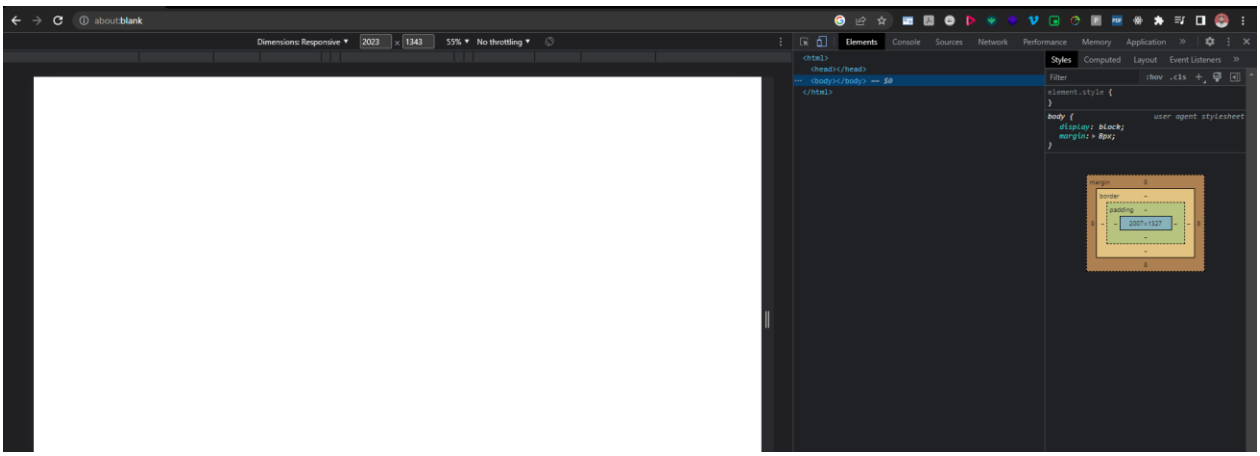
Abrimos nuestro navegador web, en este caso utilizaremos Google Chrome y abriremos una página en blanco utilizando la siguiente dirección **about:blank.**



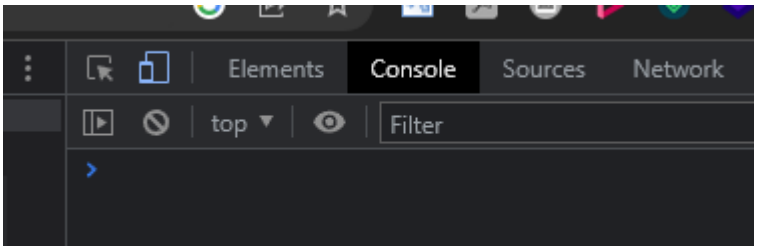
Daremos clic derecho y seleccionaremos inspeccionar.



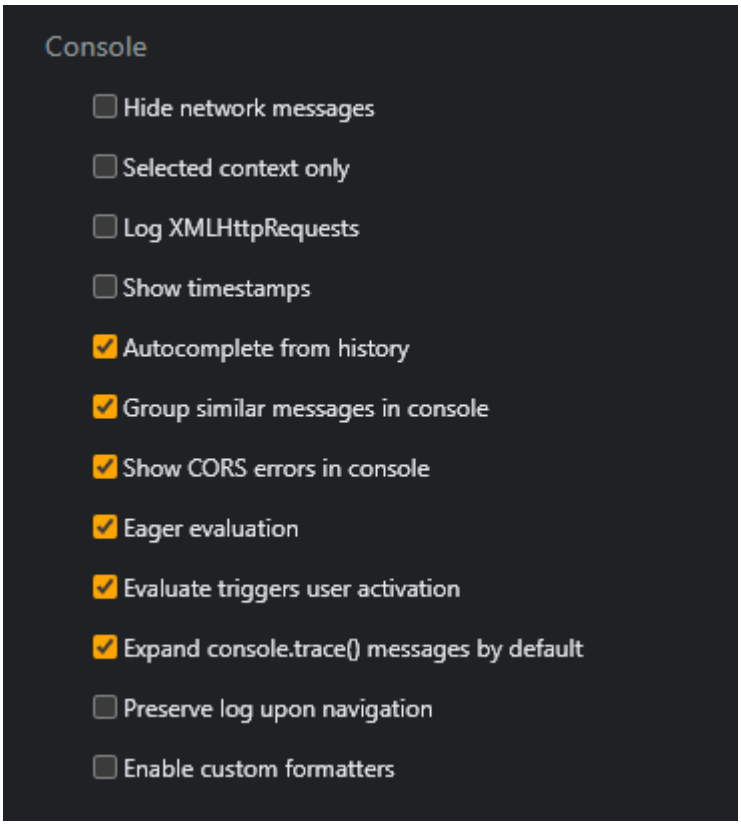
Allí nos saldrá el modo de inspección y utilizaremos las herramientas de desarrollo.



Nos dirigimos a la pestaña consola



Una vez aquí configuramos las siguientes opciones en la rueda de configuración:



Esto para tener en cuanta sobre los ejercicios que se realizaran y que nos den los resultados similares.

¿Qué es la consola?

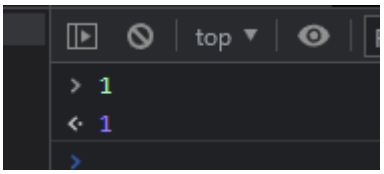
La consola es una herramienta en un entorno de desarrollo o navegador web que proporciona una forma de interactuar con un programa o sitio web a través de la línea de comandos.

En un navegador web, la consola se utiliza para depurar y realizar pruebas en un sitio web, así como para ver mensajes de error y advertencias en el código. La mayoría de los navegadores web

modernos tienen una consola integrada que se puede acceder a través de las herramientas de desarrollador.

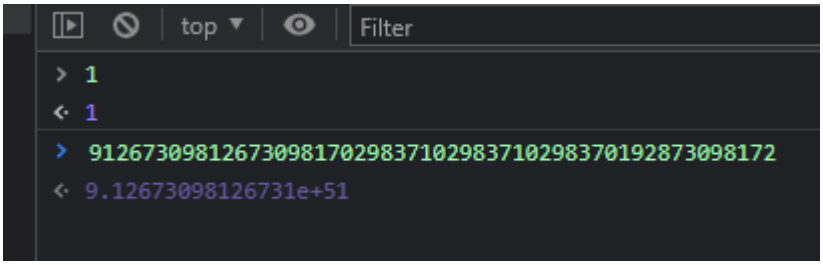
En un entorno de desarrollo, como el de una aplicación de escritorio, la consola proporciona una forma de interactuar con el programa y ejecutar comandos en tiempo real. Por ejemplo, la consola se puede utilizar para imprimir valores de variables, probar funciones o realizar otras tareas de depuración.

En resumen, la consola es una herramienta importante para los desarrolladores ya que permite interactuar y depurar código en tiempo real y ayuda a identificar y solucionar problemas en una aplicación o sitio web.

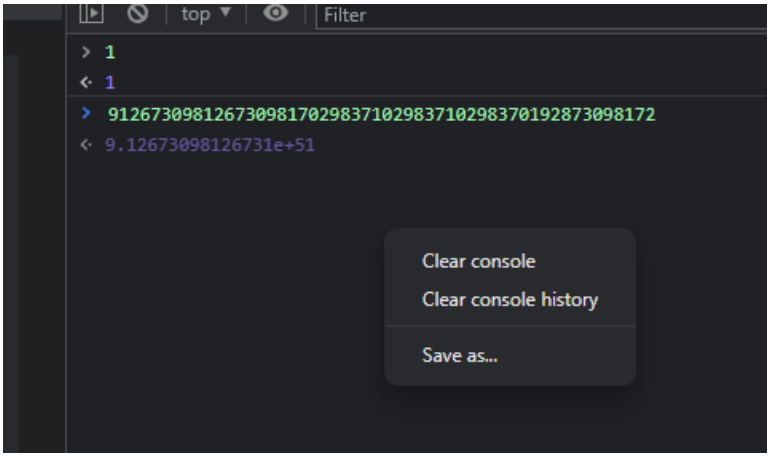


Escribiremos un numero y presionamos enter en la consola de esta manera nos mostrara un numero de ingreso y otro que nos retorna.

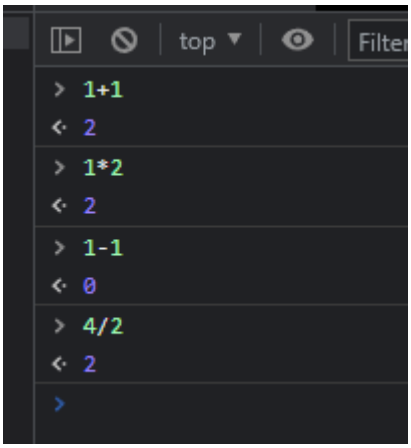
Si colocamos número al azar habrá limites como en una calculadora al representarnos ciertas cantidades de números.



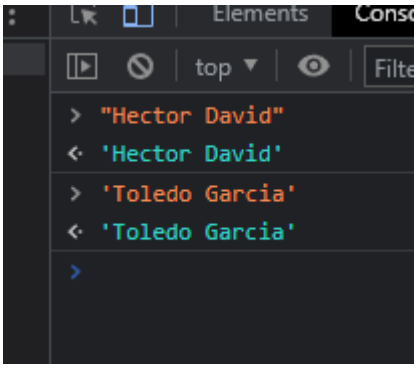
Si vemos que se nos llena de tantos números o textos la consola podremos limpiarla dándole clic derecho sobre la consola y seleccionando **clear console.**



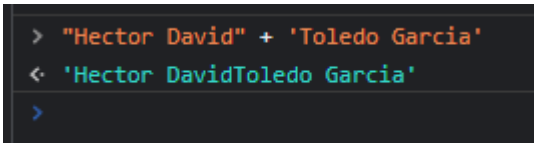
Funcionan las matemáticas en la consola de la misma manera que si tuviéramos una calculadora, todo con los operadores que conocemos.



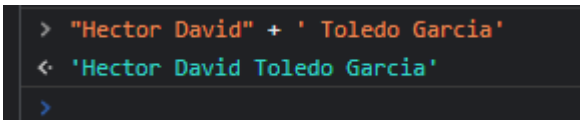
Para representar las cadenas de texto utilizamos las comillas dobles y de esta manera podremos indicarle que estamos utilizando texto, al igual que las comillas sencillas.



También podemos concatenar las cadenas de texto, utilizando el operador de suma +



Teniendo en cuenta el operador + concatenara lo que nosotros le coloquemos, es decir que para generar los espacios es necesario que nosotros agreguemos los espacios y de esta manera se nos mostrara correctamente el texto.



También se pueden utilizar de la siguiente manera:

```
> `Si una libra de arroz vale $2600, a cuanto equivalen 3 ? El valor sera de $ ${3 * 2600}`
< 'Si una libra de arroz vale $2600, a cuanto equivalen 3 ? El valor sera de $ 7800'
```

En donde aplicaremos una concatenación y realizaremos una operación interna, utilizando **${}**.

```
> `Mi edad es ${2023-1986}`
< 'Mi edad es 37'
```

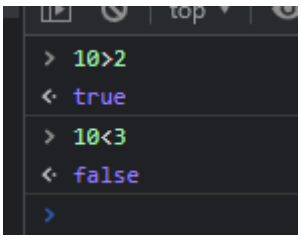
¿Qué es un booleano?

Un booleano es un tipo de dato en programación que puede tener uno de dos valores posibles: "verdadero" o "falso". En términos más técnicos, un booleano es un tipo de dato binario que representa una afirmación lógica verdadera o falsa.

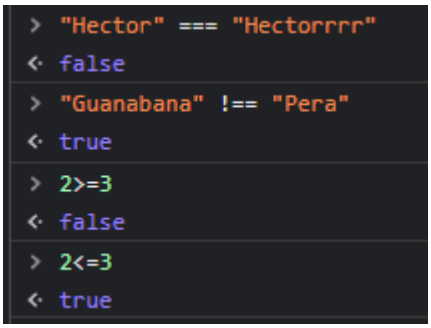
En muchos lenguajes de programación, el valor verdadero se representa con la palabra clave "true" y el valor falso se representa con la palabra clave "false". Los booleanos se utilizan comúnmente en las expresiones condicionales para tomar decisiones en el código, como en las estructuras de control de flujo, los bucles y las declaraciones de casos.

Por ejemplo, si se quisiera comprobar si un número es mayor que 10, se podría usar una expresión condicional que evalúe si la afirmación "el número es mayor que 10" es verdadera o falsa, lo que devolvería un valor booleano de "verdadero" o "falso" respectivamente.

En nuestra consola lo podremos ver de la siguiente manera:



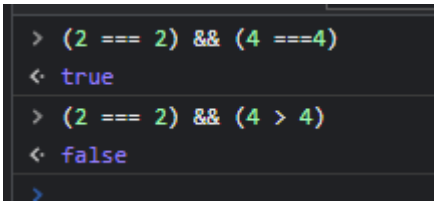
Podremos hacer comparaciones si algo es igual o no lo es:



Operadores lógicos

And &&

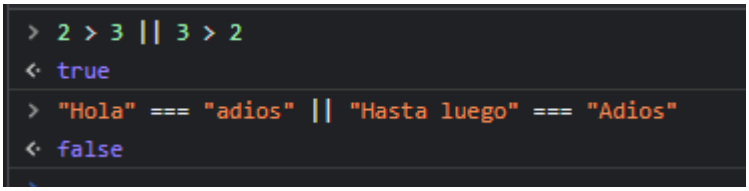
Si se cumplen ciertas series de condiciones para decir que algo es falso o verdadero:



Aquí podemos observar que en la primera opción se deben cumplir ambos criterios para que sea true y en la segunda vemos como el segundo criterio que le estamos asignando nos lo deja en false.

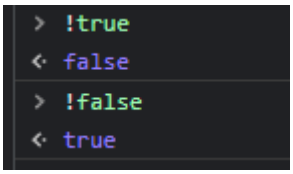
Operador OR ||

Cuando no es necesario que se cumplan ambas condiciones.



Operador Not

Es muy simple de utilizar solo anteponemos el símbolo de admiración al booleano.



Funciones

Una función en JavaScript es un bloque de código que realiza una tarea específica y que puede ser reutilizado en diferentes partes de un programa. Las funciones permiten a los desarrolladores dividir el código en piezas más pequeñas y fáciles de entender, lo que hace que el programa sea más fácil de mantener y depurar.

En JavaScript, las funciones se definen usando la palabra clave **function**, seguida de un nombre y un conjunto de paréntesis que pueden incluir parámetros separados por comas. Dentro de las llaves de la función se escribe el código que se ejecutará cuando la función sea llamada.

Por ejemplo, la siguiente función en JavaScript toma dos números como parámetros y devuelve su suma:

```
function sum(a, b) {
  return a + b;
}
```

Esta función se puede llamar en cualquier parte del programa pasando dos números como argumentos:

```
let result = sum(3, 5);
```

En este caso, la variable **result** contendría el valor **8**, que es el resultado de sumar **3** y **5**.

Variables

En JavaScript, las variables se utilizan para almacenar valores. En ES6 (también conocido como ECMAScript 2015), se introdujeron dos nuevas formas de declarar variables: **let** y **const**.

La palabra clave **let** se utiliza para declarar variables que se pueden reasignar más adelante. Por ejemplo:

```
let nombre = 'Juan';
nombre = 'Pedro';
console.log(nombre); // Output: Pedro
```

En este ejemplo, declaramos una variable llamada **nombre** y le asignamos el valor 'Juan'. Luego, reasignamos el valor a 'Pedro' y lo imprimimos en la consola. Como se puede ver, el valor de nombre ha cambiado a 'Pedro'.

Por otro lado, la palabra clave **const** se utiliza para declarar variables que no se pueden reasignar una vez que se les ha asignado un valor. Por ejemplo:

```
const pi = 3.14;
pi = 3; // Esto generará un error
```

En este ejemplo, declaramos una variable llamada pi y le asignamos el valor de 3.14. Luego, tratamos de reasignar un nuevo valor a pi lo cual no es posible debido a que fue declarado con **const**.

También es importante destacar que las variables en JavaScript se pueden declarar sin asignarles un valor inicial. Por ejemplo:

```
let edad;
console.log(edad); // Output: undefined
```

En este ejemplo, declaramos una variable llamada **edad** pero no le asignamos un valor inicial, por lo que el valor de edad es **undefined**.

Otra forma de declarar variables es mediante la utilización de la palabra clave **var**. Sin embargo, a partir de ES6, se recomienda utilizar **let** y **const** en su lugar debido a que tienen un alcance de bloque más claro y reducen la posibilidad de errores de programación.

Condiciones

Las condicionales en JavaScript se utilizan para tomar decisiones en función de ciertas condiciones. Se pueden utilizar diferentes tipos de condicionales en JavaScript, como **if**, **else if**, **else**, **switch**, entre otros.

En ES6, se introdujeron algunas mejoras en las condicionales que hacen que el código sea más legible y conciso. A continuación, explicaré algunos de los cambios más significativos:

Operador ternario

El operador ternario (?) se utiliza para escribir condicionales en una sola línea de código. Es una forma abreviada de escribir un **if** y **else**. La sintaxis es la siguiente:

```
condicion ? expresion_si_true : expresion_si_false
```

Operador de encadenamiento opcional

El operador de encadenamiento opcional **(?.)** se utiliza para acceder a una propiedad o método de un objeto solo si este está definido. Si el objeto no está definido, la expresión se evalúa como **undefined** en lugar de lanzar un error. La sintaxis es la siguiente:

```
objeto?.propiedad
objeto?.metodo()
```

Por ejemplo:

```
let persona = {
  nombre: "Juan",
  direccion: {
    calle: "Av. Libertad",
    numero: 123
  }
};

console.log(persona.direccion?.calle); // "Av. Libertad"
console.log(persona.telefono?.numero); // undefined
```

Declaración let y const en bloques

En ES6, se puede declarar una variable utilizando **let** o **const** dentro de un bloque de código (delimitado por llaves). Esto permite limitar el alcance de la variable a ese bloque, en lugar de tener un alcance global. Por ejemplo:

```
let x = 1;

if (true) {
  let x = 2;
  console.log(x); // 2
}

console.log(x); // 1
```

Ejercicios Prácticos

En estos ejercicios básicos realizaran las funciones y mostraran sus valores a través de la consola, Utilizaran ES6 y buenas prácticas en el código.

1. Crea una función que reciba dos números y devuelva el mayor de ellos.
2. Crea una función que reciba un número y devuelva true si es par y false si es impar.
3. Crea una función que reciba un número y devuelva su valor absoluto.
4. Crea una función que reciba una cadena de texto y devuelva true si la cadena tiene más de 10 caracteres y false si no.
5. Crea una función que reciba dos cadenas de texto y devuelva true si ambas cadenas tienen la misma longitud y false si no.
6. Crea una función que reciba un número y devuelva true si está entre 20 y 50 (incluyendo ambos extremos) y false si no.
7. Crea una función que reciba un objeto con dos propiedades (name y age) y devuelva un mensaje de bienvenida que incluya el nombre y la edad.
8. Crea una función que reciba un array de números y devuelva la suma de todos ellos.
9. Crea una función que reciba un array de números y devuelva true si todos son positivos y false si no.
10. Crea una función que reciba un array de cadenas de texto y devuelva true si todas tienen más de 5 caracteres y false si no.

A continuación, los ejercicios que realizaran utilizaran el **prompt,** para capturar los datos que van a preguntar y mostraran el resultado mediante un **alert.**

Aquí un ejemplo de cómo funciona:

```
// Usando prompt para pedirle al usuario su nombre
const nombre = prompt("Por favor, introduce tu nombre:");

// Usando una template literal para crear un mensaje personalizado
const mensaje = `¡Hola ${nombre}! ¿Cómo estás hoy?`;

// Usando alert para mostrar el mensaje al usuario
alert(mensaje);
```

En este ejemplo, utilizamos la función **prompt** para pedirle al usuario que introduzca su nombre. Luego, creamos un mensaje personalizado utilizando una **template literal**, que nos permite incluir variables dentro de una cadena de texto utilizando **${}**. Finalmente, utilizamos la función **alert** para mostrar el mensaje al usuario en una ventana emergente.

Ten en cuenta que es importante validar los datos que recogemos del usuario utilizando **prompt**, ya que el usuario podría introducir valores no esperados o incluso malintencionados. En general, es una buena práctica utilizar otras formas de entrada de datos más seguras, como formularios, en lugar de **prompt**.

Ejercicios:

1. Solicita al usuario su edad y muestra un mensaje que indique si es mayor de edad o no.
2. Pide al usuario que ingrese un número y muestra un mensaje que indique si es positivo o negativo.
3. Solicita al usuario su nombre y su edad y muestra un mensaje que indique si es mayor de edad o no.
4. Pide al usuario que ingrese un número y muestra un mensaje que indique si es par o impar.
5. Solicita al usuario su nombre y su género, y muestra un mensaje que indique si es hombre o mujer.

Bucles

En JavaScript, los bucles son estructuras de control que nos permiten repetir un bloque de código varias veces. Hay varios tipos de bucles que se pueden utilizar en JavaScript, y en ES6, se introdujo una nueva forma de escribir bucles llamada **for**...**of**.

1. **for loop**: Este es el bucle más común y se utiliza para repetir un bloque de código un número determinado de veces. La sintaxis es la siguiente:

```
for (let i = 0; i < 10; i++) {
  console.log(i);
}
```

   Este bucle se repetirá 10 veces e imprimirá los números del 0 al 9 en la consola.

2. **while loop**: Este bucle se utiliza para repetir un bloque de código mientras se cumpla una condición determinada. La sintaxis es la siguiente:

```
let i = 0;
while (i < 10) {
  console.log(i);
  i++;
}
```

Este bucle también imprimirá los números del 0 al 9 en la consola.

3. **do...while loop**: Este bucle se utiliza para repetir un bloque de código al menos una vez y luego seguir repitiéndolo mientras se cumpla una condición determinada. La sintaxis es la siguiente:

```
let i = 0;
do {
  console.log(i);
  i++;
} while (i < 10);
```

Este bucle también imprimirá los números del 0 al 9 en la consola.

4. **for...in loop**: Este bucle se utiliza para iterar sobre las propiedades de un objeto. La sintaxis es la siguiente:

```
const obj = { a: 1, b: 2, c: 3 };
for (const prop in obj) {
  console.log(prop + ': ' + obj[prop]);
}
```

Este bucle imprimirá en la consola los nombres de las propiedades del objeto y sus valores correspondientes.

5. **for...of loop**: Este bucle se utiliza para iterar sobre los elementos de una matriz o cualquier objeto iterable, como una cadena de texto o un mapa. La sintaxis es la siguiente:

```
const arr = [1, 2, 3, 4, 5];
for (const element of arr) {
  console.log(element);
}
```

Este bucle imprimirá los elementos de la matriz del 1 al 5 en la consola.

Ejercicios

Realizaran estos 5 ejercicios utilizando bucles con ES6, tomando la información a través del **prompt** y mostrando los resultados en un **alert**:

1. Realizar una suma de los números ingresados por el usuario hasta que ingrese un número negativo.
2. Mostrar los números del 1 al 10 en orden ascendente y luego en orden descendente.
3. Pedir al usuario una cantidad de números y mostrar cuántos son pares y cuántos impares.
4. Realizar la tabla de multiplicar del número ingresado por el usuario.
5. Pedir al usuario un número y mostrar si es primo o no.