

# NODE JS

## Clase 8





## ... Clase 8

01

Javascript

02

Programación  
estructurada

03

Conceptos  
generales



# Introducción a JavaScript

# 01

## ... Introducción a JavaScript

**JavaScript** es un lenguaje de programación ampliamente utilizado y versátil que se utiliza principalmente para agregar interactividad y funcionalidad a las páginas web

# 01

## ... Introducción a JavaScript

**JavaScript** es un lenguaje interpretado, lo que significa que no requiere un proceso de compilación previo y puede ser ejecutado directamente por el navegador

# 01

## ... Introducción a JavaScript

Una de las características más destacadas de JavaScript es su capacidad para manipular el contenido HTML y CSS de una página web en tiempo real. Esto permite a los desarrolladores crear efectos visuales dinámicos, interactuar con elementos de la página, realizar validaciones de formularios y mucho más.

# 01

## ... Introducción a JavaScript

Además de su uso en el desarrollo web, JavaScript también se ha expandido a otros ámbitos, como el desarrollo de aplicaciones móviles y de escritorio, gracias a frameworks como React Native y Electron, respectivamente. También es utilizado en servidores a través de Node.js, permitiendo a los desarrolladores construir aplicaciones de lado del servidor utilizando JavaScript.

# 01

## ... Paradigma

JavaScript es un lenguaje orientado a objetos, lo que significa que los desarrolladores pueden crear objetos y definir propiedades y métodos para interactuar con ellos. También es un lenguaje de tipado dinámico, lo que significa que no es necesario declarar explícitamente el tipo de una variable antes de usarla.

A black and white photograph of four people working together at a long table in a library. In the foreground, a woman on the left looks down at a piece of paper. Next to her, a man is focused on a laptop screen. To his right, another man is looking at a document. On the far right, a woman is also looking down at a piece of paper. Behind them are tall bookshelves filled with books.

# Programación estructurada

... **Programación Estructurada**

La programación estructurada es un enfoque de desarrollo de software que se basa en la organización lógica y estructurada del código.



# Bloques de Código

... **Bloques de Código**

Procedimientos y funciones: Los procedimientos y las funciones son bloques de código que pueden ser llamados y ejecutados en diferentes partes del programa. Permiten la reutilización de código y ayudan a dividir el programa en tareas más pequeñas y manejables.

# Ejemplo de Funciones

```
const leerPatente = (patente) => {
    patente = parseInt(prompt('Ingrese terminación patente de 0 a 9'));
    return patente;
}

const contarPatentePar = (patente,contador) => {
    if(patente%2==0 ){
        contador= contador +1 ;
    }
    return contador;
}
const contarPatenteImpar = (patente,contador) => {
    if(patente % 2 !== 0 ){
        contador= contador +1 ;
    }
    return contador;
}
```

# 03

## ... Ejemplo

Esta función recibe dos parámetros dato e índice. Se utiliza para solicitar al usuario un número a través de un prompt y almacenar ese valor en la variable dato. El parámetro índice se utiliza para proporcionar un número secuencial que se muestra en el prompt. La función luego devuelve el valor ingresado por el usuario.

## Ejemplo

```
/**  
 *  
 * @param {*} dato // valor a leer  
 * @param {*} indice // este es valor que recorre el loop  
 * @returns  
 */  
const leerDatos = (dato,indice) => {  
    dato = parseInt(prompt('Ingrese Número '+ (indice+1)));  
    return dato;  
}
```

# 03

## ... Ejemplo

La función sumarDatos recibe dos parámetros, dato y suma. Toma un valor (dato) y lo suma a otro valor acumulado (suma). La función retorna la suma actualizada.

# Ejemplo

```
/**  
 * @param {*} dato  
 * @param {*} suma  
 * @returns  
 */  
const sumarDatos = (dato,suma) => {  
    suma = suma + dato;  
    return suma;  
}
```

# 03

## ... Ejemplo

Esta función calcula el promedio de un conjunto de datos. Toma tres parámetros: promedio, suma y cantidadDatos. El parámetro suma representa la suma de todos los datos previamente calculados, mientras que cantidadDatos es el número total de datos ingresados. La función divide la suma entre cantidadDatos para obtener el promedio y luego devuelve el resultado.

# Ejemplo

```
/**  
 *  
 * @param {*} promedio  
 * @param {*} suma  
 * @param {*} cantidadDatos  
 * @returns  
 */  
const calcularPromedio = ( promedio , suma , cantidadDatos ) => {  
    promedio = suma / cantidadDatos ;  
    return promedio;  
}
```

# 03

## ... Ejemplo

La función contarMultiplos5 toma dos parámetros, dato y contMult5. Esta función se encarga de contar cuántos números ingresados por el usuario son múltiplos de 5. Si el dato es múltiplo de 5 (es decir,  $\text{dato} \% 5 == 0$ ), incrementa el valor de contMult5 en 1. Al final, la función devuelve el valor actualizado de contMult5.

# Ejemplo

```
/**  
 *  
 * @param {*} dato  
 * @param {*} contMult5  
 * @returns  
 */  
const contarMultiplos5 = ( dato, contMult5 ) => {  
    if( dato % 5 == 0 ){ // si dato % 5 == 0  
        contMult5++;  
    }  
    return contMult5  
}
```

# 03

## ... Ejemplo

La función App es la principal y muestra una aplicación que permite al usuario ingresar una cantidad determinada de datos numéricos (hasta 100 datos) y realiza varios cálculos con ellos.

# Ejemplo

```
const App = () => {
    var cantidadDatos = 100;
    var datos = [];
    var indice = 0;
    var suma = 0;
    var promedio = 0;
    var contMult5 = 0;
    var porcentajeMult5 = 0;
    //1
    datos[indice] = leerDatos(datos[indice],indice);

    while(indice<cantidadDatos && datos[indice]!== 0){
        indice++;
        datos[indice] = leerDatos(datos[indice],indice);
    }
    for (let i = 0; i < datos.length; i++) {
        //2
        suma = sumarDatos(datos[i], suma);
        // parte del 4
        contMult5 = contarMultiplos5(datos[i], contMult5);
    }
    //3
    promedio = calcularPromedio(promedio, suma, datos.length);
    //4
    porcentajeMult5 = calcularPorcentajeMultiplos5(porcentajeMult5,datos.length,contMult5);
    console.log('Promedio de datos ->' +promedio);
    console.log('Suma de datos ->' +suma);
    console.log('Porcentaje de multiplos de 5 es ->' +porcentajeMult5);
```

# 03

## ... Ejemplo

En la siguiente presentación se puede observar cómo se está invocando a la función principal

## Ejemplo

```
// Aplicación en funcionamiento;  
App();
```

# Estructuras de Control

... **Estructuras de Control**

Las estructuras de control permiten controlar el flujo de ejecución del programa

Estructuras de selección (if-else, switch): Permiten tomar decisiones basadas en condiciones.

# 04

## ... If-else, switch

Estructuras de selección (if-else, switch): Permiten tomar decisiones basadas en condiciones.

## If-else, switch

Ejemplo de estructura de control simple If

```
if(patente%2==0 ){
    contador= contador +1 ;
}
```

... **Estructuras de Control**

Estructuras de repetición (for, while, do-while): Permiten repetir un bloque de código mientras se cumpla una condición.

Estructuras de control de flujo (break, continue): Permiten controlar el flujo de ejecución dentro de bucles o condicionales.

# Ejemplo de estructura While

```
while ( patentes[contadorLectura]>-1 && contadorLectura<100 ) {  
    contadorLectura++;  
    patentes[contadorLectura] = leerPatente(patentes[contadorLectura]);  
}
```

## Ejemplo de estructura For

```
for (let i = 0; i < patentes.length; i++) {  
    console.log(patentes[i]);  
    contadorPar = contarPatentePar(patentes[i], contadorPar);  
    contadorImpar = contarPatenteImpar(patentes[i], contadorImpar);  
}
```

A black and white photograph of four students in a library. A girl on the far left is looking down at a piece of paper. Next to her, a boy is looking at a laptop screen. In the center, another boy is looking down at a piece of paper. On the right, a girl is also looking down at a piece of paper. Behind them are tall bookshelves filled with books.

# Variables Locales y Globales

... **Variables Locales y Globales**

En la programación estructurada, se prefiere el uso de variables locales, que son variables definidas dentro de un bloque o una función y solo están disponibles en ese contexto

... **Variables Locales y Globales**

Las variables globales, por otro lado, se definen fuera de cualquier bloque y pueden ser accedidas desde cualquier parte del programa. Es recomendable limitar el uso de variables globales para evitar posibles efectos secundarios y dificultades de mantenimiento.

# Ejemplo de variable local

```
function ejemploLocal() {  
    var variableLocal = "Esto es una variable local";  
    console.log(variableLocal);  
}  
  
ejemploLocal(); // Salida: "Esto es una variable local"  
  
// Si intentamos acceder a variableLocal fuera de la función, obtendremos un error  
console.log(variableLocal); // Error: variableLocal is not defined
```

# Ejemplo de variable global

```
var variableGlobal = "Esto es una variable global";

function ejemploGlobal() {
  console.log(variableGlobal);
}

ejemploGlobal(); // Salida: "Esto es una variable global"

// Podemos acceder a variableGlobal desde fuera de la función también
console.log(variableGlobal); // Salida: "Esto es una variable global"
```

# Modularización

... **Modularización**

La modularidad es un principio clave en la programación estructurada.

Consiste en dividir el programa en módulos o unidades más pequeñas y coherentes, lo cual facilita la comprensión, el mantenimiento y la reutilización del código.

# Evitar Saltos

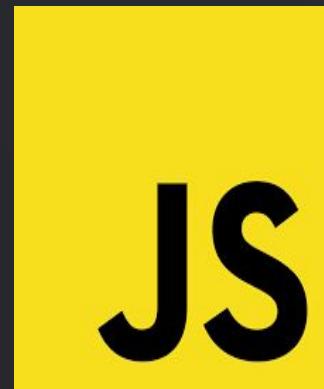
... **Evitar saltos**

Evitar saltos incondicionales: Los saltos incondicionales, como los "goto", no son utilizados en la programación estructurada debido a su naturaleza no estructurada y su impacto negativo en la legibilidad y mantenibilidad del código

# Constantes y Variables

... **Constantes y Variables**

En JavaScript, las variables y las constantes son elementos fundamentales para almacenar y manipular datos en un programa



... **Constantes y Variables**

Una variable es un contenedor que se utiliza para almacenar valores.

Puedes pensar en una variable como una caja etiquetada donde puedes guardar diferentes tipos de información, como números, texto o booleanos

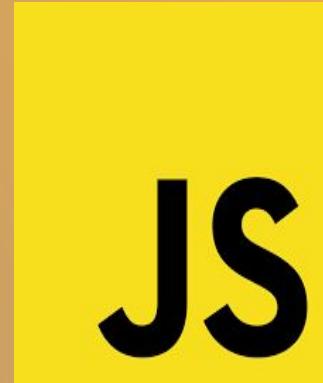
... **Constantes y Variables**

Para crear una variable en JavaScript, se utiliza la palabra clave "var", "let" o "const", seguida del nombre de la variable y, opcionalmente, se le asigna un valor inicial.

# Constantes y Variables

Ejemplo de código javascript en el que se crean tres variables diferentes con valores iniciales.

```
var dni = "36254343";
var nombre = "Maria del Valle del Pilar";
var calificacion = 9.88;
```



JS

# Constantes y Variables



Ejemplo de código javascript en el que se constante diferentes valores.

```
const registro = '26487';
const seniorityDeveloper = "Ssr";
const appName = "Comanda";
const textDescriptionApp = "Es una app que funcionará en bares y cafes";
```

JS

A black and white photograph of four people working together at a long table in a library. There are bookshelves filled with books in the background. On the table, there are laptops, notebooks, and a small container holding pens and paper. The people are focused on their work, looking down at their screens or papers.

# Tipos de datos JavaScript

... **Tipos de datos**

JavaScript es un lenguaje de programación de tipado dinámico, lo que significa que las variables no tienen un tipo de dato fijo y pueden contener diferentes tipos de datos en diferentes momentos, es decir , una variable puede tener un dato entero en un momento y después ser actualizado el valor por un dato tipo string.

... **Tipos de datos 1**

- Números (Number): Representan valores numéricos, ya sea enteros o decimales. Por ejemplo: 10, 3.14, -5.
- Cadenas de texto (String): Representan secuencias de caracteres encerrados entre comillas simples ("") o comillas dobles (''). Por ejemplo: "Hola", 'JavaScript'.
- Booleanos (Boolean): Representan los valores lógicos verdadero o falso. Los dos valores posibles son true (verdadero) y false (falso).

... **Tipos de datos 2**

- Nulos (Null): Representa la ausencia intencional de cualquier objeto o valor.
- Indefinidos (Undefined): Representa una variable que ha sido declarada pero no asignada a ningún valor.
- Objetos (Object): Son estructuras de datos que pueden contener propiedades y métodos.

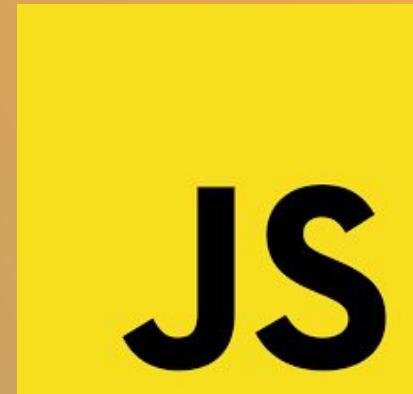
... **Tipos de datos 3**

- Arreglos (Array): Son colecciones ordenadas de elementos. Los elementos de un arreglo se almacenan en posiciones numeradas y se pueden acceder utilizando un índice. Los arreglos se definen utilizando corchetes [ ].
- Funciones (Function): Son bloques de código reutilizables que se pueden llamar y ejecutar. Las funciones en JavaScript pueden tener parámetros y devolver un valor.

# Tipos de datos

Ejemplo de código JavaScript tipos de datos

```
// Variables de diferentes tipos de datos
var numero = 10;
var cadena = "Hola, mundo!";
var booleano = true;
var nulo = null;
var indefinido;
```



JS

# Tipos de datos

Ejemplo de código JavaScript tipos de datos

```
// Imprimir los valores de las variables
console.log(numero);      // Imprime: 10
console.log(cadena);      // Imprime: Hola, mundo!
console.log(booleano);    // Imprime: true
console.log(nulo);         // Imprime: null
console.log(indefinido);   // Imprime: undefined
```



JS

# Tipos de datos

Ejemplo de código JavaScript tipos de datos

```
// Arreglo
var arreglo = [1, 2, 3, 4, 5];
console.log(arreglo);          // Imprime: [1, 2, 3, 4, 5]
console.log(arreglo[2]);        // Imprime: 3
```

JS

# Tipos de datos

Ejemplo de código JavaScript tipos de datos

```
// Objeto
var objeto = {
    nombre: "Juan",
    edad: 25,
    casado: false
};
console.log(objeto);      // Imprime: { nombre: 'Juan', edad: 25, casado: false }
console.log(objeto.nombre); // Imprime: Juan
console.table(objeto);     // Imprime: Valores del objeto en forma de tabla
```

JS

... **Primeros Desarrollos**

- Tomar 2 números por pantalla y realizar
- Operaciones básicas
- Mostrar los dos números
- Mostrar resultados

# Primeros Desarrollos



```
// Declaración de variables
var numeroA = '';
var numeroB = '';
var resultadoSuma = 0;
var resultadoResta = 0;
var resultadoMultiplicacion = 0;
var resultadoDivision = 0;
```

# Primeros Desarrollos



```
// Lectura de datos
numeroA = parseInt(prompt('Ingrese Número A : '));
numeroB = parseInt(prompt('Ingrese Número B : '));
```

# Primeros Desarrollos



```
// Operaciones

resultadoSuma = numeroA + numeroB;
resultadoResta = numeroA - numeroB;
resultadoMultiplicacion = numeroA * numeroB;
resultadoDivision = numeroA / numeroB;
```

# Primeros Desarrollos

```
// Mostrar valores y resultados
console.log('Número A: '+numeroA);
console.log('Número B: '+numeroB);

console.log('Resultado Suma : '+resultadoSuma);
console.log('Resultado Resta : '+resultadoResta);
console.log('Resultado Multiplicación : '+resultadoMultiplicacion);
console.log('Resultado División : '+resultadoDivision);
```

# ¡Muchas gracias!

