

# NODE JS

Clase 14





## ... Clase 15

01

Top-Down

02

Análisis de  
Problemas

03

Top-Down Funciones  
Top-Down

04

Ejemplos  
TopDown-Funciones

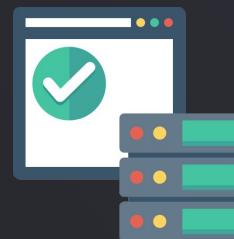
A black and white photograph of four people working at a long desk in a library. There are bookshelves filled with books in the background. On the left, a woman in a striped shirt looks down at a piece of paper. Next to her, a man in a dark jacket is looking at a laptop screen. In the center, another man with a beard is looking down at a piece of paper. On the right, a woman in a polka-dot shirt is also looking down at a piece of paper. The scene is dimly lit, with the subjects appearing as dark silhouettes against the bright background of the library.

Top-Down

# 01

## ... Top-Down

El método top-down, también conocido como enfoque descendente, es una estrategia utilizada en el desarrollo de software y la resolución de problemas en la que se comienza por analizar el problema general y luego se descompone en subproblemas más pequeños y manejables.



# 01

## ... Top-Down

Top-down implica diseñar la arquitectura general del sistema, identificar los módulos o componentes principales y luego desglosarlos en submódulos más detallados.

# 01

## ... Top-Down

La idea detrás del enfoque top-down es abordar primero los aspectos más generales y estratégicos del problema antes de profundizar en los detalles más específicos.

# 01

## ... Top-Down

El método top-down se basa en la idea de que es más fácil entender y abordar un problema cuando se tiene una visión clara y general de todo el sistema antes de profundizar en los detalles.



## ... Clase 15

01

Top-Down

02

Análisis de  
Problemas

03

Top-Down Funciones  
Top-Down

04

Ejemplos  
[TopDown-Funciones](#)



# Análisis de Problemas

... **Ejemplo 1**

Escribir un algoritmo que permita leer la cantidad de horas trabajadas por un empleado y el precio de la hora de trabajo, también que calcule y muestre el salario que le corresponde cobrar a ese empleado. Recordar que:

$$\text{Salario} = \text{cantidad de Horas} * \text{Precio de Hora trabajada.}$$

# Top-Down

```
// 1) Leer Datos
// 2) Calcular salario
// 3) Mostrar datos

    // 1.1) Leer horas de trabajo
    // 1.2) Leer valor Hora de trabajo

    // 2.1) salario total es cantidad de horas por valor hora

    // 3.1) Mostrar Salario Total
```

# Script

```
var cantidadHoras = 0;
var valorHora = 0;
var salarioTotal = 0;

valorHora = parseInt(prompt('Ingrese el valor de hora de trabajo'));
cantidadHoras = parseInt(prompt('Ingresar Canridad de Horas Trabajadas'));

salarioTotal = valorHora * cantidadHoras;

console.log('El salario total es  '+salarioTotal+'$');
```

... **Ejemplo 2**

Escribir un algoritmo que permita leer datos numéricos enteros hasta que aparezca un valor igual a cero, que calcule y muestre la suma de los datos leídos.

# Top-Down

```
// 1) Leer datos
// 2) Calcular suma de todos
// 3) Mostrar datos

    // 1.1) Leer numero
    // 1.2) Mientras numero sea distinto de cero
    // 1.3) Leer numero

    // 2.1) for numero 1 hasta N
    // 2.2) sumar números

    // 3.1) Mostrar suma de números
```

# Script

```
var numeros = [];
var suma = 0;
var contador = 0;
numeros[contador] = parseInt(prompt('Ingrese Número:'+(contador+1)));

while(contador <= 30 && numeros[contador]!== 0){
    contador++;
    numeros[contador] = parseInt(prompt('Ingrese Número:'+(contador+1)));
}

for( let i=0; i<=contador ; i++){
    suma = suma + numeros[i];
}

console.log('La suma total es -> '+suma);
```

## ... Ejemplo 3

Una inmobiliaria paga a sus vendedores un salario base, más una comisión fija por cada venta realizada, más el 5% del valor de esas ventas.

Realizar un programa que imprima el número del vendedor y el salario que le corresponde en un determinado mes. Se leen por teclado el número del vendedor, la cantidad de ventas que realizó y el valor total de las mismas.

# Top-Down

```
// 1) Leer Datos
// 2) Calcular Salario
// 3) Mostrar Datos
    // 1.1) Leer cantidad de empleados
    // 1.2) con un for
    // 1.3) Leer Numero de vendedor
    // 1.4) Leer Cantidad de ventas
    // 1.5) Leer valor total de las ventas

    // 2.1) En un for
    // 2.2) SalarioTotal es valor total de ventas * 1.05 + salario fijo;

    // 3.1) En un for
    // 3.2) Mostrar Numero de vendedor
    // 3.3) Mostrar salario total
```

# Script

```
var N =0;
var numeroVendedores = [];
var cantidadVentas = [];
var valorVentas = [];
var salarioTotal = [];
var salarioFijo =0;
N = parseInt(prompt('Ingrese Cantidad de Vendedores :'));
salarioFijo = parseInt(prompt('Ingrese salario fijo de vendedores :'));

for(let i=0; i<N; i++){
    numeroVendedores[i]= parseInt(prompt('Ingrese número de identificación de vendedor: '+ (i+1)));
    cantidadVentas[i]= parseInt(prompt('Ingrese cantidad de ventas :'+(i+1)));
    valorVentas[i] = parseInt(prompt('Ingrese valor de ventas:' +(i+1)));
}

for(let i=0; i<N ; i++){
    salarioTotal[i] = valorVentas[i]*1.05+salarioFijo;
}

for(let i=0; i<N; i++){
    console.log('Identificación de vendedor : '+numeroVendedores[i]);
    console.log('El salario total es'+salarioTotal[i]);
}
```



## ... Clase 15

01

Top-Down

02

Análisis de  
Problemas

03

Top-Down Funciones  
Top-Down

04

Ejemplos  
TopDown-Funciones

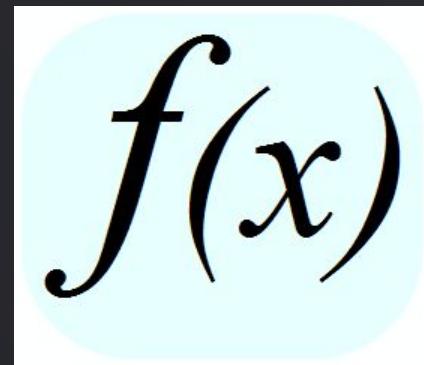


# Top-Down Funciones y Scripts

# 03

## ... Funciones

Son uno de los conceptos fundamentales de la programación y juegan un papel crucial en el lenguaje, brindando varios beneficios y ventajas.



# 03

## ... Funciones

Las funciones en JavaScript son bloques de código reutilizable que realizan una tarea específica.



# 03

## ... Funciones

A continuación, se destacan algunas de las razones por las que las funciones son importantes en JavaScript y en la programación en general.

# 03

## ... Modularidad

Las funciones ayudan a dividir un programa en módulos más pequeños y manejables. Cada función puede enfocarse en una tarea específica, lo que facilita la comprensión del código y mejora su mantenibilidad.

... Abstracción

Las funciones pueden ocultar detalles internos de su implementación al proporcionar sólo una interfaz externa para interactuar con ellas. Esto permite a los desarrolladores usar funciones sin preocuparse por los detalles internos de su funcionamiento.

# 03

## ... Mejora la legibilidad

Al dividir el código en funciones con nombres significativos, el código se vuelve más legible y fácil de entender. Las funciones bien nombradas comunican su propósito y hacen que el código sea más expresivo.

# 03

## ... Facilita la depuración

Si ocurre un error en una función, sólo es necesario revisar y corregir esa función en lugar de rastrear problemas en todo el programa. Esto simplifica la depuración y facilita la localización y corrección de errores.

# Argumentos

Las funciones pueden recibir argumentos que les permiten adaptarse y trabajar con diferentes valores. Estos argumentos se pueden utilizar para personalizar el comportamiento de la función sin necesidad de crear múltiples versiones de la misma.

```
/**  
 *  
 * @param {*} dato // valor a leer  
 * @param {*} indice // este es valor  
 * @returns  
 */  
const leerDatos = (dato,indice) => {
```

... Encapsulación

Las funciones permiten encapsular la lógica de un programa, lo que significa que variables y definiciones locales dentro de la función no interfieren con el código externo.

# 03

## ... Optimización

El uso adecuado de funciones puede mejorar el rendimiento del código, especialmente si se aplican técnicas de optimización en funciones críticas.



## ... Clase 15

01

Top-Down

02

Análisis de  
Problemas

03

Top-Down Funciones  
Top-Down

04

Ejemplos  
TopDown-Funciones

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. The people are focused on their work, looking down at papers and laptops. The scene is well-lit, with natural light coming from the side.

# Ejemplos TopDown-Funciones

... **Ejemplo**

En una bodega se cuenta con los datos sobre la cantidad de viñateros, la cantidad de uva cosechada por cada viñatero, y el código del tipo de uva (1 es moscatel, 2 es comunes, 3 es blancas y 4 es tintas). Teniendo en cuenta que cada viñatero solo cosecha un tipo de uva, se pide escribir un algoritmo que permita leer toda la información relevante de la bodega, que calcule y muestre:

... **Ejemplo**

- a) El total de uva entregada a la bodega por los viñateros.
- b) El porcentaje del conjunto de uvas moscatel y blancas sobre el total entregado.
- c) La cantidad de kilos de uvas tintas cosechadas.

## Ejemplo

```
// 1) Lectura de datos
// 2) Calcular total de uvas entregadas por los viñateros
// 3) Calcular porcentaje de uvas moscatel y blancas sobre el total entregado
// 4) Calcular cantidad de kg de uvas tintas.
// 5) Mostrar datos.
```

## Ejemplo

```
// 1.1 Leer cantidad N de datos  
// 1.2 Leer Código de Uva N veces  
// 1.3 Leer Cantidad de Kg N veces
```

## Ejemplo

```
// 2.1 Sumar N cantidad de Kg  
// 2.2 Inicializar una variable suma en cero  
// 2.3 Hacer un bocle For  
// 2.4 Sumar todos los datos de Kg
```

## Ejemplo

```
// 3.1 Contar Uvas Blancas
// 3.2 Contar Uvas Moscatel
// 3.3 Porcentaje de Uvas Blancas es Cantidad de uvas blancas / total
// 3.4 Porcentaje de Uvas Moscatel es Cantidad de uvas Moscatel / total
```

# Ejemplo



```
// 4.1 Sumar Uvas Tintas
```

## Ejemplo

```
// 5.1 Mostrar Total de kg  
// 5.2 Mostrar Porcentaje de Uvas Blancas  
// 5.3 Mostrar Porcentaje de Uvas Moscatel  
// 5.4 Mostrar Cantidad de kg de Uvas Tintas
```

# Ejemplo

```
// Definición de Funciones
/**
 *
 * @param {*} dato  tipo de Uva y también Cantidad de Kg
 * @param {*} i      Variable de recorrido de array
 * @param {*} mensaje Mensaje a mostrar para el usuario
 */
const leerDatos = ( dato, i,mensaje) => {
    dato =parseInt(prompt(mensaje+ (i+1)+ ' '));
    return dato;
}
```

## Ejemplo

```
/**
 *
 * @param {*} N      // Cantidad de Datos
 * @param {*} kgUvas // Kg de uvas
 * @param {*} sumaKg // suma de kg
 * @returns
 */
const sumarkilogramos = ( kgUvas , sumaKg ) => {
    sumaKg = sumaKg + kgUvas;
    return sumaKg;
}
```

# Ejemplo

```
/**
 *
 * @param {*} tipoUvas
 * @param {*} contarTipoUvas
 * @param {*} uvaBuscada
 * @returns
 */
const contarTipoUvas = (tipoUvas, contarTipoUvas, uvaBuscada) => {
    if(tipoUvas === uvaBuscada){
        contarTipoUvas++;
    }
    return contarTipoUvas;
}
```

# Ejemplo

```
/**
 *
 * @param {*} contarTipoUvas contador de tipo de uva buscado
 * @param {*} N cantidad total de uvas
 * @param {*} porcentaje porcentaje
 * @returns
 */
const calcularPorcentajeUvas = ( contarTipoUvas , N , porcentaje ) => {
    porcentaje = (contarTipoUvas/N)*100;

    return porcentaje;
}
```

# Ejemplo

```
/**  
 *  
 * @param {*} tipoUva          Tipos de uvas  
 * @param {*} sumaUvasTintas  Suma de uvas tintas  
 * @param {*} kgUvas           kilogramos de uvas  
 * @returns  
 */  
const sumarTintas = (tipoUva , sumaUvasTintas , kgUvas ) => {  
    if(tipoUva === 4){  
        sumaUvasTintas = sumaUvasTintas + kgUvas;  
    }  
    return sumaUvasTintas;  
}
```

## Ejemplo

```
/**  
 *  
 * @param {*} dato           Función que muestra dato ingresado  
 * @param {*} mensaje        Mensaje a mostrar antes de la variable  
 */  
const mostrarDatos = (dato,mensaje) => {  
    console.log(mensaje+dato);  
}
```

## Ejemplo

```
// Declaración de Variables
var N = 0;
var tipoUvas = [];
var kgUvas = [];
var sumaTotalKg =0;
var contadorUvasMoscatel =0;
var contadorUvasBlancas =0;
var porcentajeMoscatel =0;
var porcentajeBlancas =0;
var sumarUvasTintas =0;
```

## Ejemplo

```
// 1 Lectura de Datos
N = parseInt(prompt('Ingrese cantidad de viñateros : '));

for(let i=0; i<N; i++){
    tipoUvas[i] = leerDatos(tipoUvas[i],i,'Ingrese tipo de Uvas');
    kgUvas[i] = leerDatos(kgUvas[i], i, 'Ingrese cantidad de kilogramos de uvas');
}
```

## Ejemplo

```
// 2 Procesamiento de Datos
for (let i = 0; i < N; i++) {
    sumaTotalKg = sumarkilogramos(kgUvas[i], sumaTotalKg);
    contadorUvasBlancas = contarTipoUvas(tipoUvas[i],contadorUvasBlancas,3);
    contadorUvasMoscotel = contarTipoUvas(tipoUvas[i],contadorUvasMoscotel,1);
    sumarUvasTintas = sumarTintas(tipoUvas[i], sumarUvasTintas,kgUvas[i]);
}

porcentajeBlancas = calcularPorcentajeUvas(contadorUvasBlancas,N, porcentajeBlancas);
porcentajeMoscotel= calcularPorcentajeUvas(contadorUvasMoscotel,N, porcentajeMoscotel);
```

## Ejemplo

```
mostrarDatos(sumaTotalKg,'Suma total de kg: ');
mostrarDatos(porcentajeBlancas+' %','Porcentaje de uvas Blancas: ');
mostrarDatos(porcentajeMoscateL+' %', 'Porcentaje de Uvas Moscatel:');
mostrarDatos(sumarUvasTintas, 'Kilogramos de uvas tintas :');
```

# ¡Muchas gracias!

