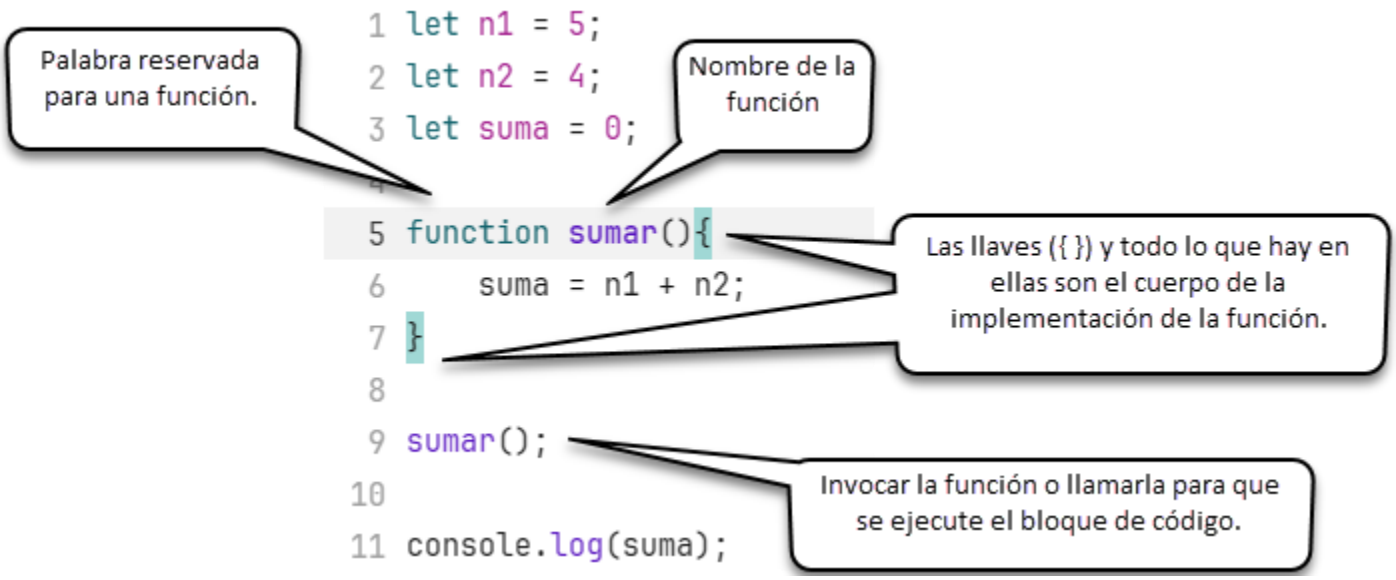
Funciones en JavaScript

Una función es un conjunto de instrucciones que realiza una tarea o calcula un valor.




```
let n1 = 5;
let n2 = 4;
let suma = 0;

function sumar(){
    suma = n1 + n2;
}

sumar();

console.log(suma);
```

función con retorno

```
function calcularMayor(){
    let a = 5;
    let b = 2;
    return a > b? `${a} es el mayor`: `${b} es el mayor`;
}

console.log(calcularMayor());
```

Al ejecutar la función retorna o devuelve un valor.

Ejemplo de función

```
function calcularTabla(){
    let numeroTabla = 2;
    mostrar = ``;
    for (let i = 1; i ⩽ 5; i++) {
        mostrar+= `${i} x ${numeroTabla} = ${ i * numeroTabla}\n`;
    }
    return mostrar;
}

console.log(calcularTabla());
```

Función con parámetros

Parámetros son parte de la definición de una función. se separan por ",".

```
function mostrarMensaje(nombre, mensaje){
    return `Hola ${nombre}, ${mensaje}`;
}

console.log(mostrarMensaje("Juan","Bienvenido a algoritmos!"));
```

Argumentos son parte de la llamada de una función.

```
let saldo = 12000;
let opcion = 1;
function mostrarMenu(menu){
    return `${menu}`;
}

function mostrarSaldo(){
    return `su saldo es ${saldo}`;
}

function retirar(retiro){
    saldo-=retiro;
    return `su retiro es de ${retiro}, nuevo saldo ${saldo}`;
}

while(opcion != 3){
    opcion = Number(prompt(mostrarMenu("1. mostrar saldo \n 2. retirar \n 3. salir")));
    switch(opcion){
        case 1: alert(mostrarSaldo()); break;;
        case 2: alert(retirar(Number(prompt("valor a retirar:"))));break;
    }
}
```

El retorno de la función se asigna a una constante o variable

```
const x = function mostrar(a, b, c){
    return a + b + c;
}
console.log(x(3, 4, 5));
```

La inmobiliaria "Sucasa"

```
let venta = 0;
function determinarComision(tipo) {
    if(tipo == 1) comision = 0.10;
    else if(tipo == 2) comision = 0.12;
    return comision;
}

function mostrarmensaje(comision){
    return `La parte del cliente es ${venta-=venta* comision} y la parte de la inmoviliaria es ${venta*comision}`
}

venta = prompt("ingrese el valor de la venta:")
alert(mostrarmensaje(determinarComision(Number(prompt("1. Inmueble usado 2. Inmueble nuevo")))));
```

Lava Ya

```
precio = 0;
function tipoLavado(tLavado){
    if(tLavado == "1") return 2000;
    else if(tLavado == "2") return 15000;
    else return 0;
}

function seleccionarServicio(tipo) {
    switch (tipo) {
        case "1":{
            alert(`El servicio de lavado vale ${tipoLavado(prompt("1. En seco 2. Normal")) * Number(prompt("cantidad prendas:"))}`);
            break;
        }
      case "2":{
          alert(`El servicio de planchado ${Number(prompt("cantidad de prendas: ")) * 1800 + 3000}, se cobro domicilio `);
          break;}
      default:
          break;
    }
}

seleccionarServicio(prompt("1. Lavado 2. planchado →"));
```

stikers

```
function costoStikers(tamaño, cantidad){
 return `Costo de los estikers: cantidad=> ${cantidad} tiene un costo de $
 {cantidad * tamaño}`
}

alert(costoStikers(Number(prompt("1 pequeño 2 Grande")) == 1? 4000 : 6000 ,
Number(prompt("ingrese la cantidad: "))));
```

Nombre de la declaración de la función.

parámetros

```
1    function ordinary1(a, b, c) {
2        // implementación de la función
3    }
4
5    const ordinary2 = function (a, b, c)
6        return a + b + c;
7    };
```

Las llaves ({ }) y todo lo que hay entre ellas son el cuerpo de la declaración de la función.

instrucción devuelve explícitamente un valor de la función

Rol desempeñado por las funciones

```
1  function add(x, y) {
2      return x + y;
3  }
4   console.log(add( x: 2, y: 3));
5
6  const objeto = { metodo1 : add }
7
8  console.log(objeto.metodo1( x: 4, y: 5));
```

invocada a través de una llamada de la función.

almacenado en una propiedad

invocado a través de una llamada de método.

Funciones de flecha

```
1  const f = function (x, y, z) { return x };
2
3  const f1 = (x, y, z) => { return x };
4
5  console.log(f( x: 1, y: 5, z: 8));
6  console.log(f1( x: 1, y: 5,  z: 8));
7
```

La función de flecha (más o menos) equivalente se ve de la siguiente manera. Las funciones de flecha son expresiones.

Se elimina los { } y el return

Si una función de flecha tiene un solo parámetro y ese parámetro es un identificador , puede omitir los paréntesis alrededor del parámetro.

```
1  const f = (x, y, z) => x;
2
3  const f1 = x => x;
4
5  console.log(f( x: 1, y: 5, z: 8));
6  console.log(f1( x: 2));
7
```

Usamos los paréntesis para decirle a JavaScript que el cuerpo es una expresión (un objeto literal) y no un bloque de código.

```
1  const func1 = () => ({a: 1});
2
3  const func2 = () => {a: 1};
4
5  console.log(func1(1,5,8));
6  console.log(func2(2));
```

```
{ a: 1 }
undefined
```

Manejo de parámetros

Los parámetros son parte de una definición de función.

```
1  const sum = (x, y) => x + y;
2
3  console.log(sum( x: 3, y: 6))
```

Los argumentos son parte de una llamada de función

Los valores predeterminados de los parámetros especifican el valor que se utilizará si no se ha proporcionado un parámetro.

```
1  function f(x, y :number = 5 ) {
2      return [x, y];
3  }
4
5  console.log(f( x: 1))
6
```

De dos parámetros solo proporcionamos 2 argumentos