Estructura de Programa

Pablo Ortiz G.

¿Qué es?

En este capítulo, comenzaremos a hacer cosas que realmente se pueden llamar *programación*. Expandiremos nuestro dominio del lenguaje JavaScript más allá de los sustantivos y fragmentos de oraciones que hemos visto hasta ahora, al punto donde podemos expresar prosa significativa, para lo cual debemos reconocer los siguientes términos.

Expresión:
Un fragmento de código que produce un valor,
Ejemplo: fragmento de una Oración

Vinculaciones o (Variables, Constantes):
Espacio en memoria para atrapar y mantener valores.

Ejemplo:

```
let salario = 500*30;
```

Palabras Reservadas: `break case catch class const continue debugger default delete do else enum export extends false finally for function if implements import interface in instanceof let new package private protected public return static super switch this throw true try typeof var void while with yield`

Declaración:
Un Bloque de código que produce un valor
Ejemplo: Oración completa

Fuente: Eloquent JavaScript, Thirt Edition - Marijn Haverbeke

Instructor: Pablo Ortiz G.

Términos Clave

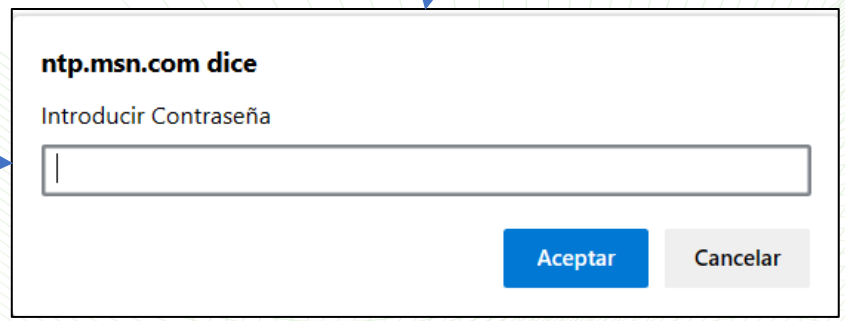
Entorno: La colección de vinculaciones y sus valores que existen en un momento dado se llama *entorno*. Cuando se inicia un programa, este entorno no está vacío. Siempre contiene vinculaciones que son parte del estándar del lenguaje, y la mayoría de las veces, también tiene vinculaciones que proporcionan formas de interactuar con el sistema circundante.

Función: es una pieza de programa envuelta en un valor. Dichos valores pueden ser *aplicados* para ejecutar el programa envuelto. Por ejemplo, en un entorno navegador, la vinculación prompt sostiene una función que muestra un pequeño cuadro de diálogo preguntando por entrada del usuario.

Ejemplo:

```
prompt("Introducir contraseña");
```

Valores de Retorno: Mostrar un cuadro de diálogo o escribir texto en la pantalla es un *efecto secundario*. Muchas funciones son útiles debido a los efectos secundarios que ellas
producen. Las funciones también pueden producir valores, en cuyo caso no necesitan tener un efecto secundario para ser útil..



Fuente: Eloquent JavaScript, Thirt Edition - Marijn Haverbeke

Instructor: Pablo Ortiz G.

Estructuras de Control

Flujo Lineal

Las estructuras de control son el conjunto de reglas que dan el poder de alterar, controlar o modificar el orden o el flujo en el que se ejecutan las instrucciones de un software (Programa) a voluntad. Gracias a las estructuras de control se puede abstraer algoritmos o secuencias de instrucciones en un software para lograr su objetivo.

JavaScript admite un compacto conjunto de declaraciones, específicamente declaraciones de control de flujo, que puedes utilizar para incorporar una gran cantidad de interactividad en tu aplicación.

Flujo de control: Cuando tu programa contiene más de una declaración, las declaraciones se ejecutan como si fueran una historia, de arriba a abajo.

Flujo Estándar de JS

```
let Numero1 = Number(prompt("Elige un numero"));
let Numero2 = Number(prompt("Elige un numero"));
let resultado = Numero1 + Numero2;
console.log("La suma de los numero es: " + resultado);
```

```
La suma de los numero es: 11
```

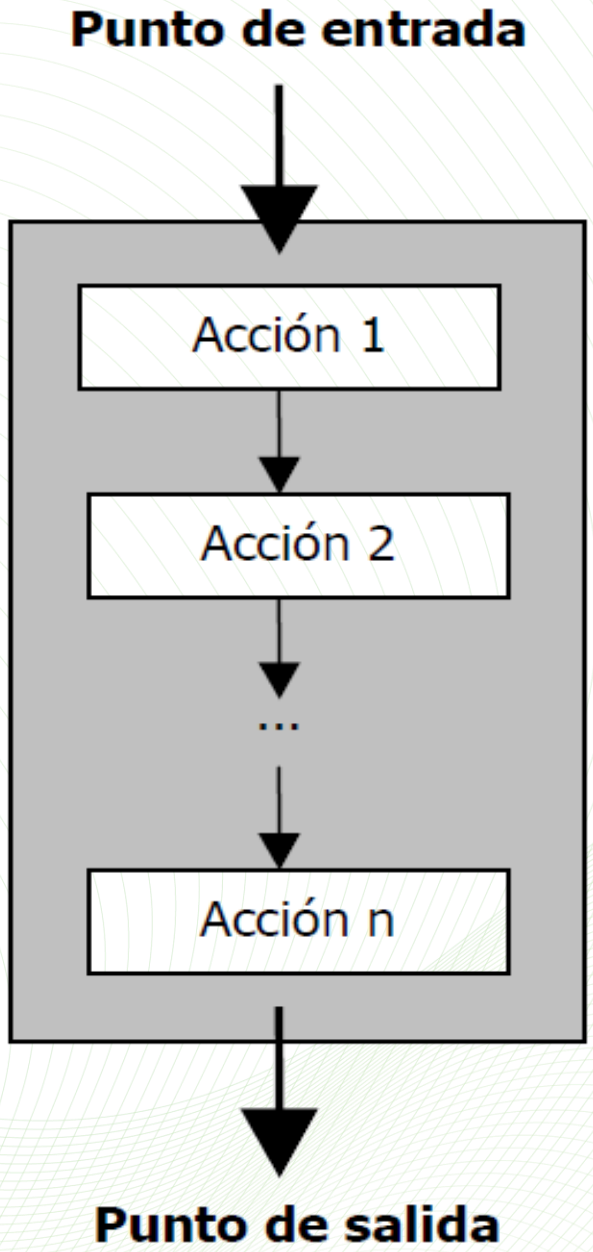
Fuente: Eloquent JavaScript, Thirt Edition - Marijn Haverbeke

Instructor: Pablo Ortiz G.

Estructura de Bloque

La sentencia de bloque es el tipo de sentencia más básico y se utiliza para realizar agrupaciones de sentencias, es decir nos permite delimitar un conjunto de acciones englobando las mismas dentro de un bloque compacto. Para delimitar el bloque de sentencias se han de utilizar las llaves “{ }”, al principio y al final de bloque.

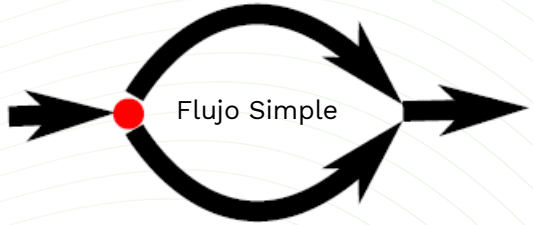
```
// Ejercicios y Prácticas de Programación JavaScript
// Ejemplo de ámbito de bloque
// Declaración de variable
var numero = 10;
var numerodos = 20;
{
    var numero = 20;
    let numerodos = 40;
}
console.log(numero);
console.log(numerodos);
```



Fuente: Eloquent JavaScript, Thirt Edition - Marijn Haverbeke

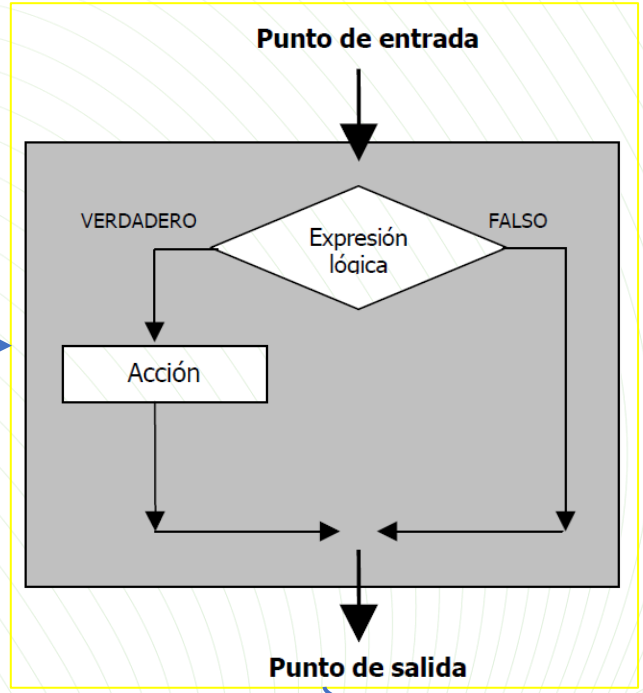
Instructor: Pablo Ortiz G.

Estructura Selectiva Simple IF



Estructura alternativa simple: Esta estructura permite evaluar una expresión lógica y en función de dicha evaluación ejecutar una acción (o composición de acciones) o no ejecutarla; también se la suele denominar SI-ENTONCES.

Si la condición se cumple (es decir, si su valor es true) se ejecutan todas las instrucciones que se encuentran dentro del bloque {...}. Si la condición no se cumple (es decir, si su valor es false) no se ejecuta ninguna instrucción contenida en {...} y el programa continúa ejecutando el resto de las instrucciones del script.




```
var mostrarMensaje = true;

if(mostrarMensaje) {
  console.log("Hola Mundo");
}
```

```
var mostrado = false;
var usuarioPermiteMensajes = true;

if(!mostrado && usuarioPermiteMensajes) {
  console.log("Es la primera vez que se muestra el mensaje");
}
```

```
var mostrarMensaje = true;

if(mostrarMensaje == true) {
  console.log("Hola Mundo");
}
```

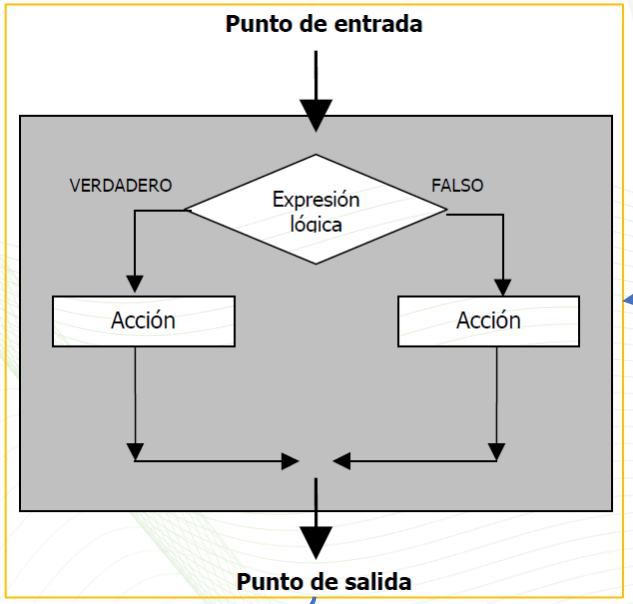
```
var mostrado = false;

if(!mostrado) {
  console.log("Es la primera vez que se muestra el mensaje");
}
```

```
if (condicion) {
    expresion 1...;
    expresion 2...;
    ..........;
}
```

Fuente: Eloquent JavaScript, Thirt Edition - Marijn Haverbeke

Instructor: Pablo Ortiz G.

Estructura Selectiva Doble if



Estructura alternativa doble: La estructura alternativa doble es similar a la anterior con la salvedad de que en este tipo de estructura se indican acciones no sólo para la rama “verdadera” sino también para la “falsa”; es decir, en caso de la expresión lógica evaluada sea cierta se ejecutan una acción o grupo de acciones y en caso de que sea falsa se ejecuta un grupo diferente. (Si entonces)

A menudo no solo tendrás código que se ejecuta cuando una condición es verdadera, pero también código que maneja el otro caso. Con la palabra reservada **else (if, else)**

```
SI (A > B) ENTONCES
        Escribir "El Mayor es A"
SINO
        Escribir "El Mayor es B"
FIN_SI
```

```
if (condicion) {
    expresion 1....;
    expresion 2....;
    ..........;
} else{
    expresion 1....;
    expresion 2....;
    ..........;
}
```

```
var edad = 18;

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

```
let numero = Number(prompt("Elige un numero"));

if (numero < 10) {
console.log("Pequeño");
} else if (numero < 100) {
console.log("Mediano");
} else {
console.log("Grande");
}
```


Fuente: Eloquent JavaScript, Thirt Edition - Marijn Haverbeke

Instructor: Pablo Ortiz G.

Ejercicios

1. Escribe un programa que pida al usuario un número y muestre si es par o impar.
2. Escribe un programa que pida al usuario dos números y muestre el mayor de ellos.
3. Escribe un programa que pida al usuario un año y muestre si es bisiesto o no.
4. Escribe un programa que pida al usuario una letra y muestre si es vocal o consonante.
5. Escribe un programa que pida al usuario un mes y muestre el número de días que tiene.
6. Escribe un programa que pida al usuario una calificación numérica y muestre su equivalente en letras (A, B, C, D o F).
7. Escribe un programa que pida al usuario una temperatura en grados Celsius y muestre su equivalente en grados Fahrenheit.
8. Escribe un programa que pida al usuario una contraseña y muestre si es correcta o no. La contraseña válida es “1234”.
9. Escribe un programa que pida al usuario un color y muestre si es uno de los colores del arcoíris o no.
10. Escribe un programa que pida al usuario un animal y muestre si es un mamífero, un ave, un reptil, un anfibio, un pez o un invertebrado.

Resolver usando la estructura condicional anidada:

1. Escribe un programa que pida al usuario un número y muestre si es positivo, negativo o cero.
2. Escribe un programa que pida al usuario tres números y muestre el mayor, el menor y el medio de ellos.
3. Escribe un programa que pida al usuario un día de la semana y muestre si es laboral o no.
4. Escribe un programa que pida al usuario un mes y muestre la estación del año a la que pertenece.
5. Escribe un programa que pida al usuario una edad y muestre si es menor de edad, mayor de edad o jubilado.
6. Escribe un programa que pida al usuario un carácter y muestre si es una letra mayúscula, una letra minúscula, un número o un símbolo.
7. Escribe un programa que pida al usuario dos números enteros y muestre si son divisibles entre sí o no.
8. Escribe un programa que pida al usuario una nota numérica y muestre su equivalente en letras según el siguiente criterio: A (10-9), B (8-7), C (6-5), D (4-3), F (2-0).

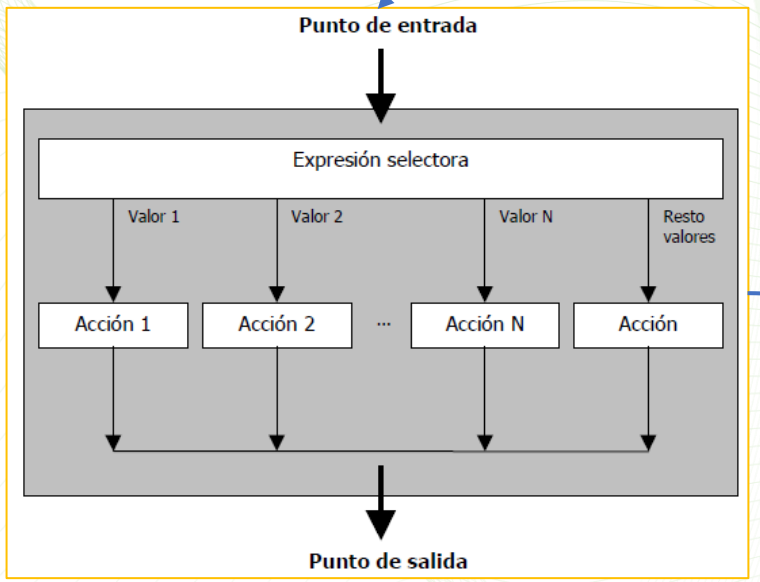
Ejercicios de Javascript: los condicionales If, swtich (espaciolatino.com)

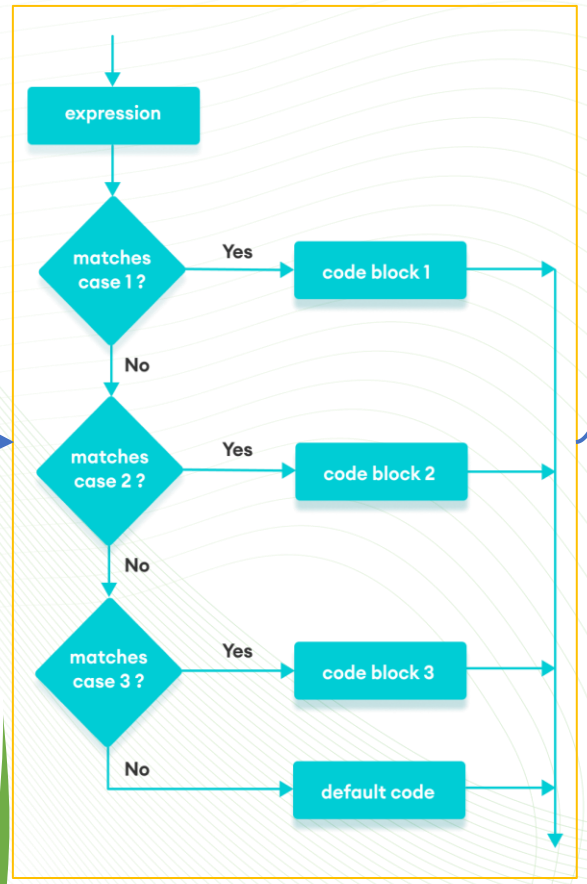
Fuente: Eloquent JavaScript, Thirt Edition - Marijn Haverbeke

Instructor: Pablo Ortiz G.

Estructura Selectiva Switch

La declaración de JavaScript se utiliza en la toma de decisiones. Evalúa una expresión y ejecuta el cuerpo correspondiente que coincide con el resultado de la expresión (En caso de que la condición sea)






```
switch(variable/expression) {
    case valor1:
        // Cuerpo para el caso 1
        break;

    case valor2:
        //  Cuerpo para el caso 2
        break;

    case valorN:
        // Cuerpo para el caso N
        break;

    default:
        // Cuerpo para el caso contrario
}
```

```
let a = 2;

switch (a) {
    case 1:
        a = 'one';
        break;
    case 2:
        a = 'two';
        break;
    default:
        a = 'not Existe';
        break;
}
console.log(`El Valor es: ${a}`);
```

Puede poner cualquier cantidad de etiquetas de case dentro del bloque abierto por switch. El programa comenzará a ejecutarse en la etiqueta que corresponde al valor que se le dio a switch, o en default si no se encuentra ningún valor que coincida. Continuará ejecutándose, incluso a través de otras etiquetas, hasta que llegue a una declaración break.

Fuente: Eloquent JavaScript, Thirt Edition - Marijn Haverbeke

Instructor: Pablo Ortiz G.

1.Calificación de letras: Escribe un programa que tome una letra como entrada y devuelva un mensaje según la letra ingresada (A, B, C, D, F).
2.Días de la semana: Crea un programa que tome un número del 1 al 7 y devuelva el día correspondiente de la semana.
3.Meses del año: Escribe un programa que tome un número del 1 al 12 y devuelva el nombre del mes correspondiente.
4.Operaciones matemáticas básicas: Desarrolla un programa que tome dos números y un operador (+, -, *, /) y realice la operación correspondiente.
5.Traductor de idiomas: Crea un programa que tome una palabra en inglés y la traduzca a otro idioma utilizando **switch** para diferentes casos.
6.Calculadora de tarifa de envío: Desarrolla un programa que calcule la tarifa de envío de acuerdo al peso del paquete y el destino.
7.Conversor de unidades: Escribe un programa que convierta entre diferentes unidades de medida (por ejemplo, centímetros a pulgadas, kilogramos a libras, etc.).
8.Detección de tipo de dato: Crea un programa que tome un valor y determine si es un número, una cadena de texto, un booleano o un objeto.
9.Generador de mensajes personalizados: Desarrolla un programa que tome un evento como entrada y devuelva un mensaje personalizado dependiendo del evento (cumpleaños, boda, graduación, etc.).
10.Menú de opciones: Implementa un programa que muestre un menú con diferentes opciones y realice una acción correspondiente según la opción seleccionada.

Fuente: Eloquent JavaScript, Thirt Edition - Marijn Haverbeke

Instructor: Pablo Ortiz G.

Estructura Repetitiva Ciclos While

El flujo de control de ciclos nos permite regresar a algún punto del programa en donde estábamos antes y repetirlo con nuestro estado del programa actual.

La palabra while es seguida por una expresión en paréntesis y luego por una declaración, muy similar a if. El bucle sigue ingresando a esta declaración siempre que la expresión produzca un valor que dé true cuando sea convertida a Boolean. (Mientras)

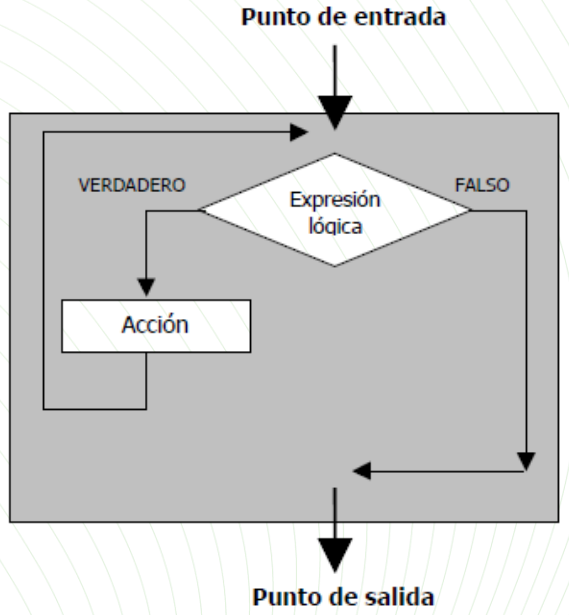
```
let resultado = 1;
let contador = 0;

while (contador < 10) {
resultado = resultado * 2;
contador = contador + 1;
}

console.log(resultado);
```

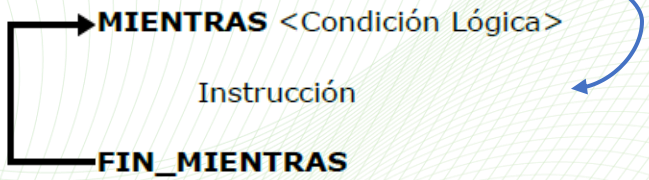
```
let numero = 1;
let listaNumeros = [];

while (numero <= 10) {
  listaNumeros.push(numero);
  numero++;
}
```

```
let numero = 1;

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

Esta estructura While repite una acción o grupo de acciones mientras una expresión lógica sea cierta;

```
while (condicio){
    instrucciones;
};
```





Atajos similares:
*= 2 para duplicar resultado.
+= 1 contar hacia adelante.
-= 1 para contar hacia abajo.

Fuente: Eloquent JavaScript, Thirt Edition - Marijn Haverbeke

Instructor: Pablo Ortiz G.

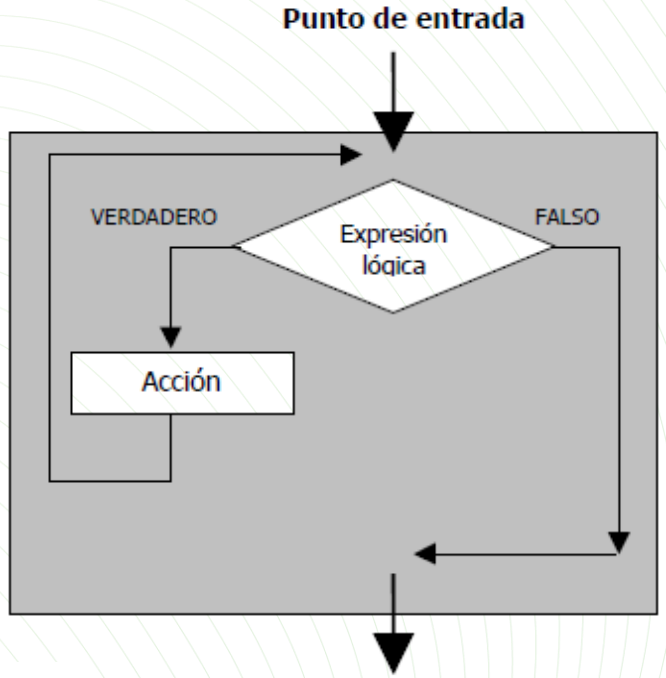
Estructura While

1.Suma de números: Escribe un programa que sume todos los números enteros positivos desde 1 hasta un número ingresado por el usuario utilizando un bucle **while**.
2.Contador regresivo: Crea un programa que imprima una cuenta regresiva desde un número ingresado por el usuario hasta 1 utilizando un bucle **while**.
3.Factorial: Desarrolla un programa que calcule el factorial de un número ingresado por el usuario utilizando un bucle **while**.
4.Adivina el número: Implementa un juego donde el usuario debe adivinar un número aleatorio generado por el programa. Utiliza un bucle **while** para repetir el proceso hasta que el usuario adivine el número correcto.
5.Generador de secuencia Fibonacci: Escribe un programa que genere los primeros N números de la secuencia Fibonacci utilizando un bucle **while**.
6.Contador de dígitos: Crea un programa que cuente y muestre la cantidad de dígitos en un número ingresado por el usuario utilizando un bucle **while**.
7.Tabla de multiplicar: Desarrolla un programa que imprima la tabla de multiplicar de un número ingresado por el usuario utilizando un bucle **while**.
8.Conversor de temperatura: Implementa un programa que convierta una temperatura en grados Celsius a grados Fahrenheit utilizando un bucle **while** para permitir al usuario realizar múltiples conversiones.
9.Detección de números primos: Escribe un programa que determine si un número ingresado por el usuario es primo o no utilizando un bucle **while**.
10.Simulador de ahorro: Crea un programa que simule un plan de ahorro, donde el usuario ingresa un monto de dinero a ahorrar mensualmente y una tasa de interés. Utiliza un bucle **while** para calcular el saldo acumulado después de un número de meses especificado por el usuario.

Instructor: Pablo Ortiz G.

Estructura Repetitiva Ciclos Do

Un ciclo DO es una estructura de control similar a un ciclo while. Difiere solo en un punto: un ciclo do siempre ejecuta su cuerpo al menos una vez, y comienza a chequear si debe detenerse solo después de esa primera ejecución. Para reflejar esto, la prueba aparece después del cuerpo del ciclo:




```
do {
  // código que se ejecuta al menos una vez
} while (condición);
```

```
let nombre;

do {
nombre = prompt("Quien eres?");
} while (!nombre);

console.log(nombre);
```

```
let sum = 0;
let i = 1;

do {
  sum += i;
  i++;
} while (i <= 10);

console.log("La suma es:", sum);
```

```
let num;

do {
  num = prompt("Ingrese un número mayor que 100:");
} while (num <= 100);

console.log("El número es:", num);
```

HACER

Instrucción...

MIENTRAS < Condición Lógica >

Fuente: Eloquent JavaScript, Thirt Edition - Marijn Haverbeke

Instructor: Pablo Ortiz G.

1. Sumar los números del 1 al 10 utilizando un bucle.
2. Pedir al usuario que ingrese un número mayor que 100 utilizando un bucle.
3. Generar un número aleatorio entre 1 y 10 hasta que se obtenga un 7.
4. Imprimir los números pares del 0 al 20.
5. Pedir al usuario que ingrese un número entre 1 y 5.
6. Imprimir los primeros 10 números de la serie de Fibonacci.
7. Pedir al usuario que ingrese una contraseña y repetir la solicitud hasta que ingrese una contraseña válida (por ejemplo, "contraseña123").
8. Imprimir los primeros 5 números primos.
9. Pedir al usuario que ingrese un número positivo utilizando un bucle **do-while** y mostrar un mensaje de error si no lo hace.
10. Imprimir los números del 10 al 1

Instructor: Pablo Ortiz G.

Estructura Repetitiva Ciclos for

Debido a que los ciclos son un patrón muy común, JavaScript y otros lenguajes similares proporcionan una forma un poco más corta y más completa, el ciclo for

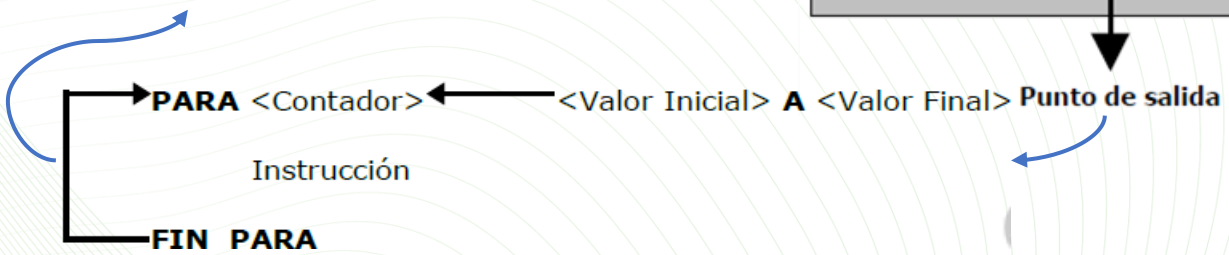
```
for (let i = 1; i <= 10; i++) {
  console.log(i);
}
```

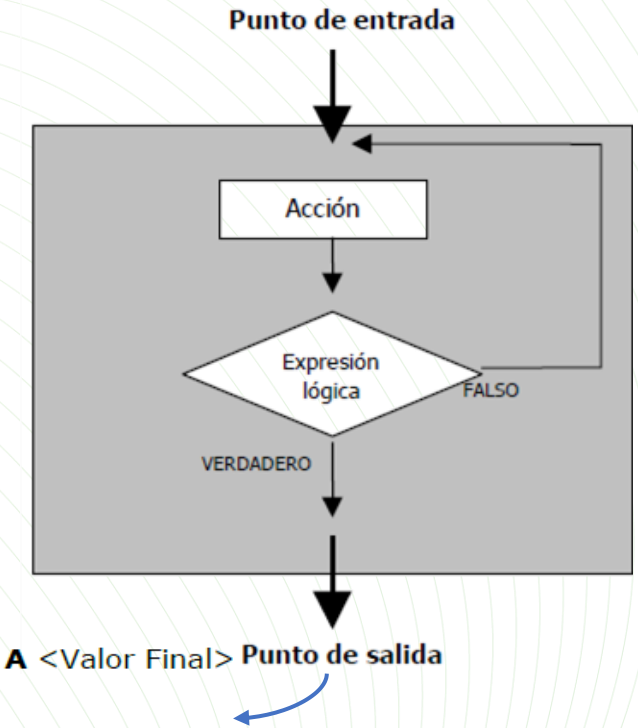
```
let sum = 0;
for (let i = 1; i <= 10; i++) {
  sum += i;
}
console.log("La suma es:", sum);
```

```
for (let i = 2; i <= 20; i += 2) {
  console.log(i);
}
```

```
let num = 5;

for (let i = 1; i <= 10; i++) {
  console.log(`${num} x ${i} = ${num * i}`);
}
```

```
for (condicion){
    instrucciones..;
}
```





El único cambio es que todos las declaraciones que están relacionadas con el "estado" del ciclo están agrupadas después del for. Los paréntesis después de una palabra clave for deben contener dos punto y comas. La parte antes del primer punto y coma *inicializa* el ciclo, generalmente definiendo una vinculación. La segunda parte es la expresión que *chequea* si el ciclo debe continuar. La parte final *actualiza* el estado del ciclo después de cada iteración. En la mayoría de los casos, esto es más corto y conciso que un constructor while o do.

Fuente: Eloquent JavaScript, Thirt Edition - Marijn Haverbeke

Instructor: Pablo Ortiz G.

1. Calcular el factorial de un número dado.
2. Imprimir los primeros 5 números primos.
3. Imprimir la secuencia de números del 1 al 100, pero para múltiplos de 3 imprimir "Fizz", para múltiplos de 5 imprimir "Buzz" y para múltiplos de ambos 3 y 5 imprimir "FizzBuzz“.
4. Imprimir los números del 10 al 1 en orden descendente.
5. Imprimir los primeros 10 términos de la serie de Fibonacci.

Fuente: Eloquent JavaScript, Thirt Edition - Marijn Haverbeke

Instructor: Pablo Ortiz G.

G R A C I A S

Línea de atención al ciudadano: 01 8000 910270
Línea de atención al empresario: 01 8000 910682

@SENAComunica

www.sena.edu.co