# Funciones y Ciclos

## Ciclos

{desafío} latam_

***Codificar un programa utilizando funciones para la reutilización de código acorde al lenguaje Javascript.***

- Unidad 1:
  Introducción al lenguaje JavaScript
- Unidad 2:
  Funciones y Ciclos
- Unidad 3:
  Arrays y Objetos
- Unidad 4:
  APIs

***Te encuentras aquí***

{desafío} latam_

# ¿Qué aprenderás en esta sesión?

- *Codificar una rutina JavaScript, aplicando las diferentes formas de declarar funciones para resolver el problema planteado.*
- *Desarrollar algoritmos utilizando ciclos de instrucciones if/else y ciclos anidados para resolver un problema de baja complejidad.*

{desafío} latam_

Según lo aprendido hasta ahora, ¿Qué contiene la estructura de una función?

# ¿Cuál es el alcance de las variables declaradas let y var?

{desafío} latam_

# ¿Para qué sirve la función Arrow?

{desafío} latam_

# /* Rutinas Repetitivas v/s Ciclos */

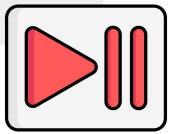
# Rutinas Repetitivas v/s Ciclos

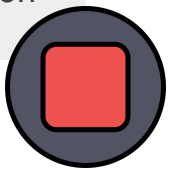
## *Ciclos*

- Los ciclos en programación nos permiten llevar a cabo tareas que son repetitivas, como por ejemplo, calcular el promedio de notas para cada estudiante.

- Un ciclo en programación suele tener 3 elementos para realizar su ejecución:

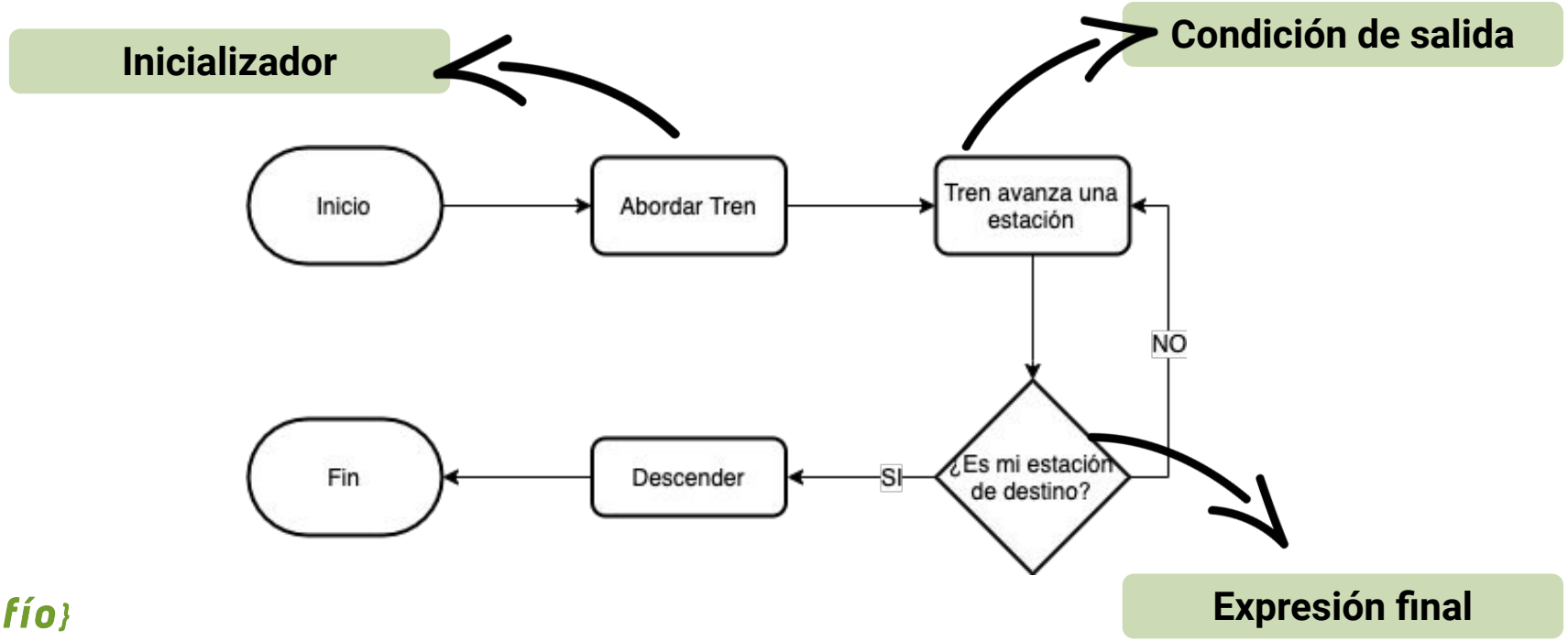
| Inicializador | Condición de salida | Expresión final |
|---|---|---|
| Usualmente es una variable asignada a un número para definir un parámetro de inicio del ciclo, suele ser conocido también como contador. | Corresponde a un Boolean, que define si se debe continuar con el ciclo o se debe detener. | Corresponde a una expresión u operación que se ejecuta al final del bloque del ciclo, suele ser una operación que acerca el contador a la condición de salida. |





# Rutinas Repetitivas v/s Ciclos

*Diagrama de Flujo*



# /* Ciclos y Estructuras de Control */

# Ciclos y Estructuras de Control

- Existen diferentes tipos de estructuras de control, para en este capítulo nos enfocaremos en las relacionadas con ciclos.

- Ya conocemos la estructura básica **for,** con sus 3 componentes, los cuales sirven para definir los parámetros de iteración de ciclo.

- Existen otras 2 palabras reservadas para alterar el funcionamiento “normal” de un ciclo: **break** y **continue.**

# Ciclos y Estructuras de Control

## *Break*

- **Break** en español significa quebrar y efectivamente eso es lo que logra en un ciclo, desde el punto (línea) donde se escribe esta palabra, el ciclo se interrumpe, quiebra o finaliza.

```
for (let i = 1; i < 10; i++) {
    if (i == 3) {
      break;
    }
    console.log(i);
}
```

El resultado en la consola del navegador será el siguiente:

```
1
2
```

# Ciclos y Estructuras de Control

## *Continue*

- De manera análoga **continue** (continuar en español), nos permite ignorar los comandos siguientes en el bloque de código del ciclo sin terminar la iteración completa sino solo el segmento actual.

```
for (let i = 1; i < 10; i++) {
    if (i == 3) {
      continue;  // se ignora la sentencia
'console.log()'
    }
    console.log(i);
}
```

El resultado en la consola del navegador será el siguiente:

```
1
2
4
5
6
7
8
9
```

# Demostración - “Estructura de control y repetitivas”

# Ejercicio guiado

## *Estructura de control y repetitivas*

El ejercicio consiste en mostrar los números pares y contar los números impares mediante un ciclo for para los números comprendidos entre el 0 y el 20 (incluidos ambos), pero cuando se llegue al número 10, el programa debe continuar y no mostrar el número en cuestión, igualmente, cuando se llegue al número 19 el ciclo debe ser interrumpido y terminar su ejecución. Para ello, sigamos los siguientes pasos:

- **Paso 1:** En el archivo script.js debes armar la estructura del ciclo repetitivo “for” para que pueda realizar el conteo desde el número 0 hasta el número 20. Quedando de la siguiente forma:

```
for (let i = 0; i <= 20; i++) {
    //cuerpo del ciclo repetitivo
}
```

# Ejercicio guiado

## *Estructura de control y repetitivas*

- **Paso 2:** Posteriormente utilizamos estructuras de control if-else, verificando si el número es par (mostrar) de lo contrario impar (contar), o es 10 o es 19 para ejecutar la acción correspondiente, como por ejemplo, en el caso de ser 10 no se debe mostrar el número, utilizando la sentencia "continue" logramos este objetivo. Para terminar la ejecución del ciclo cuando se esté en el número 19, con la sentencia "break" se puede detener abruptamente el ciclo.

```
var impar = 0;

for (let i = 0; i <= 20; i++) {
    if (i == 10) {
      continue;  // se ignora
cualquier otro proceso
    };
    if (i == 19) {
      break;  // se rompe el ciclo
actual
    };
    if (i % 2 == 0) {
      document.write(i+"<br>");
    }else {
      impar++;
    };
};
document.write("La cantidad de números
impares es: "+impar);
```

{desafío} latam_

# /* Ciclos anidados */

# Ciclos anidados

- Se conocen como ciclos anidados cuando la declaración de un ciclo es otro ciclo.
- Generalmente usamos ciclos anidados cuando usamos operaciones un poco más complejas, veamos el siguiente ejemplo.

- Busquemos todas las combinaciones entre el elemento a y b:

```
for(let i = 0; i < 3; i++) {
   for(let j = 0; j < 3; j++) {
        console.log(i,j);
    }
}
```

**¿Cuál es el resultado?**

# Demostración - “Factoriales”


{desafío}
latam_

# Ejercicio guiado

## *Factoriales*

Desarrollar un programa utilizando JavaScript que permita encontrar el factorial del 1 al 10 implementando anidación de ciclos repetitivos. Sigamos los pasos a continuación:

- **Paso 1:** Separar el problema en partes. Lo primero que haremos será calcular el factorial de un número, analizando la fórmula podemos darnos cuenta que lo más apropiado es usar un for para resolver el problema.

```
var res = 1;
for (let i = 1; i <= 5; i++) {
    res = res * i;
}
console.log(res);
```

# Ejercicio guiado
## *Factoriales*

- **Paso 2:** Hemos generado el código para calcular el número factorial de 5, pero nosotros necesitamos obtener los primeros 10 factoriales y para eso necesitamos repetir el código anterior 10 veces, así que utilizaremos un ciclo dentro de otro.

```
for (let i = 1; i <= 10; i++) {
   let res = 1;
   for (let j= 1; j <= i; j++) {
      res = res * j;
  }
  console.log("Factorial de " + i + " es: " + res);
}
```

# Demostración - “Elementos comunes en dos listados”

# Ejercicio guiado

## *Elementos comunes en dos listados*

Desarrollar un programa utilizando JavaScript que permita encontrar los elementos comunes entre dos listados de números realizados con ciclos repetitivos anidados, uno va del 1 al 5 y el segundo del 1 al 10. Por consiguiente:

- **Paso 1:** Vamos a crear dos ciclos for anidados, uno del 1 al 10 y el otro del 1 al 20 y así mostrar ambas variables utilizadas para el conteo de cada ciclo.

```
for(let i = 0; i < 5; i++) {
     console.log("i: "+i);
   for(let j = 0; j < 10; j++) {
     console.log("j: "+j);
   }
}
```

{desafío} latam_

# Ejercicio guiado

## *Elementos comunes en dos listados*

- **Paso 2:** Ejecutamos el código anterior y obtendremos lo siguiente:

```
i=0
j=1
j=2
j=3
...
...
...
j=9
i=2
j=0
j=1
...
```

# Ejercicio guiado

## *Elementos comunes en dos listados*

- **Paso 3:** Ahora si debemos buscar los elementos que sean iguales entre ambas listas, para ello basta con preguntar mediante una estructura repetitiva si las variables de cada ciclo for son igual (ciclos repetitivos for anidados).

```
for(let i = 0; i < 5; i++) {
   for(let j = 0; j < 10; j++) {
       if(i == j) {
            console.log("El número "+ i +" se encuentra en ambos
listados")
       }
    }
}
```

¿Existe algún concepto que no hayas comprendido?

Volvamos a revisar los conceptos que más te hayan costado antes de seguir adelante

# Próxima sesión...

- *Desarrollaremos juntos el material sincrónico que corresponde a un **Desafío guiado,** con el cual pondrás a prueba tus conocimientos adquiridos.*

{desafío}
latam_

{desafío}
latam_

Academia de
talentos digitales