

Bucles

Bucles: uno de los pilares de la programación.

Se creó un nuevo archivo 'bucle.js' y se remplazó en "index.html".

¿Cómo funciona este pilar?

Comenzamos a partir de una declaración, un valor asignado, la cual seguirá una secuencia siempre y cuando cumpla con una condición, si la cumple saca true y mantiene la secuencia las veces necesarias, si no cumple, sacará false y directamente finaliza el algoritmo.

Los bucles nos ayudan

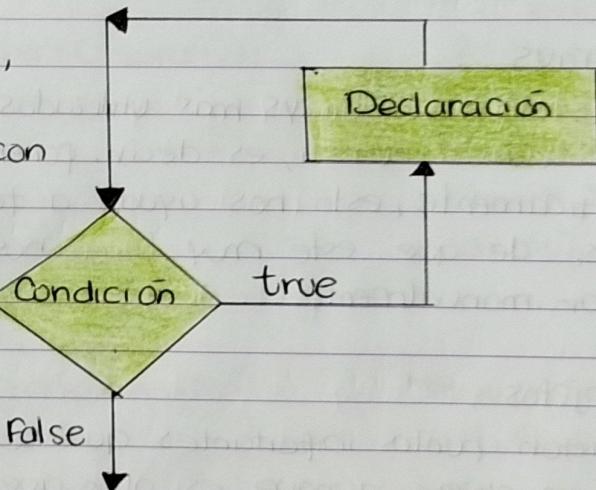
bastante cuando queremos ejecutar varias líneas de código, con la programación tradicional tendría que analizar dato por dato, pero en este caso se puede seleccionar un rango de datos, ingresarlos a este algoritmo y puede seleccionar los elementos que diga la condición hasta que lo rompa.

Código Esta estructura se divide en tres partes. Consola

```
//for
for(let i=0; i<= 5; i++){
    console.log(i);
}
```

//while

```
let i = 0;
while(i<8){
    console.log(i);
    i++;
}
```



0	
1	
2	
3	
4	
5	
6	
7	

```
// do while
let i = 0;
do {
    i = i + 2;
    console.log(i);
} while (i < 20);
```

Confirma

2
4
6
8
10
12
14
16
18
20

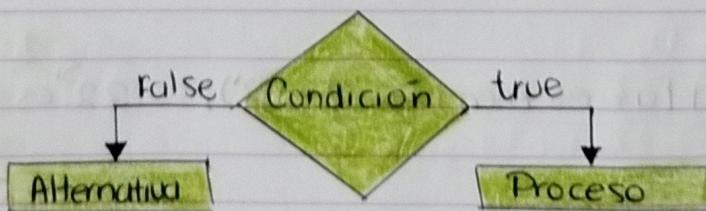
¿Cuando debes utilizar cada una de las maneras?

Eso depende de como quieras extraer la información. Si necesitas que la condicional sea lo último en revisar, entonces utiliza do-while. En realidad se toma énfasis en el último valor.

Tanto while como for, siempre comienzan desde el punto de inicio pero son lo que más necesitarás en la práctica.

Condicionales

Se creó el archivo nuevo 'condicional.js' y se remplazó el antiguo. Otro pilar fundamental dentro de la programación.



Un algoritmo donde dependiendo de una condición, va a elegir verdad o falso. Si la condición se cumple, se va por true y se almacena un proceso, en caso de no cumplirse, se va por false porque a la vez es una alternativa cuando no cumple la condición.

Con JavaScript vamos a crear la condición.

Se verán cuatro tipos:

1. Condicional IF
2. Condicional ELSE
3. Condicional ELSE-IF
4. Condicional SWITCH

Se comenzará con la número 1:

Código.

```
var a = 10;
```

```
if (a > 5) {
```

```
    console.log('SI ES MAYOR')
```

```
}
```

```
//Condicional else
```

```
var a = 3;
```

```
if (a > 5) {
```

```
    console.log('SI ES MAYOR A 5')
```

```
}
```

```
else {
```

```
    console.log('ES MENOR A 5')
```

```
}
```

```
var a = 22;
```

```
var b = 20;
```

```
if (a > b) {
```

```
    console.log(`#${a} si es mayor a: ${b}`);
```

```
}
```

```
else {
```

```
    console.log(`#${a} es menor a: ${b}`); Si 22 es mayor a: 20
```

```
}
```

```
var a = 80;
```

```
var b = 120;
```

```
if (a > b) {
```

```
    console.log(`#${a} si es mayor a: ${b}`);
```

```
}
```

```
else {
```

```
    console.log(`#${a} es menor a: ${b}`); 80 es menor a: 120
```

Concola

SI ES MAYOR

ES MENOR A 5

// Else if

var a = 80;

var b = 80;

if (a > b) {

 console.log(`\\$ {a} es mayor a: \\$ {b}`);

}

else if (a == b) {

 console.log(`\\$ {a} y \\$ {b} son iguales`);

}

else {

 console.log(`\\$ {a} es menor a: \\$ {b}`);

}

Concola

a y b son iguales

Operadores lógicos

• && = AND

Darleve verdadero si todas las condiciones se cumplen, si no, falso.

• || = OR

Darleve verdadero al menos una condición verdadera, si no, falso.

var x = true;

Concola

var y = false;

false

console.log(x && y);

var x = true;

true

var y = false;

console.log(x && y);

true

var x = false;

var y = false;

console.log(x && y);

false

// Condicional Switch

let curso = "python";

switch (curso) {

 case 'java':

 console.log("java");

 break;

Case 'python':

```
Console.log("python3")  
break;
```

Case 'javascript':

```
console.log("javascript");  
break;
```

default:

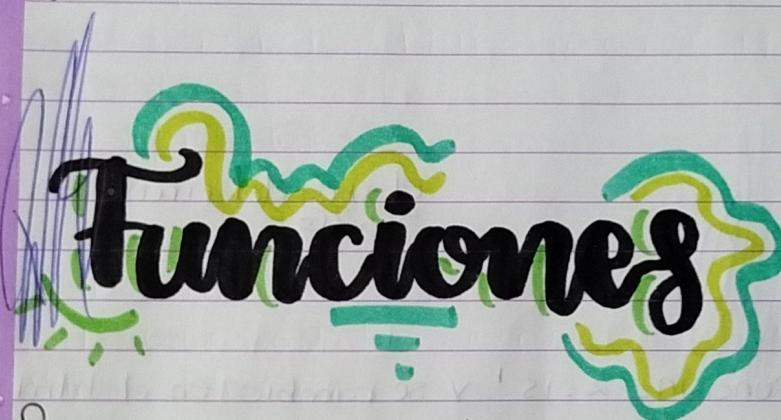
```
console.log("no disponible"); → no disponible  
break
```

}

Consola

python3

Si en lugar de buscar
'python', buscara
'php'. ya que no
se encuentra.



Se creó un nuevo archivo 'Función.js' y se remplazó por el anterior.
¿Qué es una función en JavaScript?

Es un bloques de código que se ejecutará solo cuando sea llamado.

Estructura:

```
function NOMBRE(argumento) {
```

declaración ↗ Indica que valores lleva

}

Código:

```
function Suma(a, b){
```

var sum = a + b;

```
console.log('la suma es: ' + sum)
```

función con argumentos

```
suma(8,4);
```

Consola

la suma es: 12

Se verán cuatro tipos de funciones:

1. Función con argumentos

2. Función con retorno

3. Función de flecha

4. Función anidada.

// Funciones retornables

```
function dato_trabajador() {
```

```
    var salario = 2550;
```

```
    console.log('su salario es: ' + salario);
```

```
}
```

```
var obrero = dato_trabajador();
```

Confola

su salario es: 2550

```
function dato_trabajador() {
```

```
    var salario = 2550;
```

```
    return salario;
```

```
}
```

```
var obrero = dato_trabajador();
```

```
console.log(obrero);
```

2550

// función de Flecha (anónimas)

```
var resta = function(n1, n2) {
```

```
    return n1 - n2;
```

```
}
```

```
console.log(resta(8, 2));
```

6

// simplificada.

```
var resta = (n1, n2) => n1 - n2;
```

```
console.log(resta(8, 2));
```

6

// otro ejemplo

```
var nombre = function() {
```

```
    return "Juan";
```

```
}
```

```
console.log(nombre());
```

Juan

// utilizando la estructura

```
var nombre = () => "Victor";
```

```
console.log(nombre());
```

Victor

¿Qué es una función Anónima (flecha)?

Es una función que no tiene específicamente un nombre, pero generalmente se usa como un valor basado a otra función que realmente ya lo puede ver como un valor.

Función Anidada: es una función dentro de otra función.

Código:

Consola:

// Función anidada

```
function operacion() {
```

```
    const PI = 3.1416;
```

```
    function area(radio) {
```

```
        var area = PI * radio * radio;
```

```
        console.log('el área del círculo es: ' + area);
```

```
}
```

```
    operacion.area = area;
```

```
}
```

```
var radio = 4;
```

```
operacion();
```

```
operacion.area(radio);
```

el área del círculo es:
50.26

funciones II

Se creó un nuevo archivo 'funcion_2.js' y se cambió en el html Código para añadir elementos a la página (parte visual).

```
<body>
```

```
    <h1> javascript desde cero </h1>
```

```
    <h2> área de un círculo </h2>
```

```
    <br>
```

```
    <h2> Ingrese el valor del radio </h2>
```

```
    <input type="text" id="radio">
```

```
    <h2> resultado </h2>
```

```
    <input type="button" onclick="area()" value="AREA">
```

```
    <input type="text" id="resultado">
```

```
    <form>
```

```
        <script src="funcion_2.js"></script>
```

```
<body>
```

```
</html>
```

Código para la función de operación del área. (acción)

```
function area() {
    const PI = 3.14;
    var radio = document.getElementById("radio").value;
    var resultado = PI * radio * radio;
    document.getElementById("resultado").value = resultado;
}
```

Funciones particulares

Eval: evalúa una expresión.

```
//Eval
var a = 10;
var b = 20;
var x = eval("a+b");
console.log(x);
var y = eval("3+4");
var z = eval("a+8");
var respt = a + z;
console.log(respt);
//parseFloat
var a = parseFloat("10");
var b = parseFloat("20curso");
var c = parseInt("10curso");
console.log(a);
console.log(b);
console.log(c);
//date.parse
// 1 enero de 1970;
var dato = "1 january, 1970 1:30 PM";
var resultado = Date.parse(dato);
console.log(resultado);
```

63000000
(milisegundos)

Métodos

Se creó un nuevo archivo 'método.js' y se cambió en "index.htm". El método es muy parecido a una función, simplemente que el método generalmente se aplica a un objeto, o como en este caso, a un array. Vamos a llamar al método como una propiedad de tipo función.

Temas que se abordarán:

1. Método foreach
2. Método some, every.
3. Método Map, Filter, Reduce.

Código

//foreach: una manera simple de hacer ejercicios con funciones

```
let numero = [1, 2, 3, 4, 5, 6, 7];  
numero.forEach((value) => {  
    console.log(value == 5);  
})
```

• foreach

Es una manera simplificada de hacer ejercicio con funciones.

Es un método que aplicaremos a un array. Tendremos resultados de tipo booleano. Se usa para una función interna dentro del array.

//every

```
let numeros = [5, 5, 5];  
console.log(numeros.every((value) => {  
    return (value == 5);  
}))
```

• Every

Es lo contrario a some.

Si cumple la función de false.

Para que salga true todos los elementos deben cumplir la condición.

• Some

Verifica si un valor dentro del array satisface una condición.
Itera a través de elementos. Acepta siempre expresiones booleanas

Se usa para verificar que el dato a encontrar satisfacer
nuestra necesidad.

//map

```
let num = [1, 2, 3, 4, 5, 6];  
let duplicar = num.map ((value) => {  
    return value * 2;
```

• Map

Verificar cada elemento del array a detalle.

Crea un nuevo array con el ya existente. Duplica el valor.

Devuelve a un nuevo array.

//Filter

```
let number = [1020, 3390, 1046, 2550, 1090];  
let numeros_grandes = number.filter ((value) => {  
    return value > 2500;  
});
```

• Filter

Filtrar los elementos y crea un subarray.

Crea un desglose del original.

Filtrar cada elemento a partir de un límite/condición.

//reduce

```
let numbers = [1, 2, 3, 4, 5, 6, 7];  
let respuesta = numbers.reduce ([suma, dato_actual] =>  
    suma + datos_actual, c);  
console.log(respuesta);
```

• Reduce

Encuentra valores y genera otros elementos.

Como un ejemplo de suma, va a sumar todos los datos del array.

Conclusiones

1. Los bucles permiten la repetición o interacción de fragmentos de código (nos permiten ejecutar un bloque de código varias veces seguidas).
2. Las estructuras condicionales permiten elegir entre la ejecución de una acción o otra.
3. En conclusión las funciones nos permiten dividir nuestro código en módulos más pequeños y manejables.
4. Un método consiste en una serie de sentencias para llevar a cabo una acción.