INSTITUTO KHIPU

Semestre IV

Sesión 07

PROGRAMA DE ESTUDIOS

DESARROLLO DE SISTEMAS DE INFORMACIÓN

COMUNITY MANAGER Y MARKETING DIGITAL

Tema

BUCLES EN JAVASCRIPT (FOR, WHILE, DO-WHILE)

Bucles en JavaScript (for, while, do-while)

1. Concepto de bucles y su utilidad en programación

Los bucles (o ciclos) son estructuras de control que permiten repetir un bloque de código varias veces según una condición específica. Su utilidad en programación incluye:

- Automatizar tareas repetitivas sin necesidad de escribir múltiples líneas de código.
- Recorrer estructuras de datos como arrays o listas.
- Controlar la ejecución de procesos hasta que se cumpla una condición.

En JavaScript, los bucles más utilizados son:

- `for`: Se usa cuando conocemos el número exacto de iteraciones.
- `while`: Se usa cuando la repetición depende de una condición lógica.
- `do-while`: Similar a `while`, pero garantiza al menos una ejecución del bloque de código.

2. Estructura y uso de los diferentes tipos de bucles

2.1 Bucle `for`: Iteraciones controladas con una variable de contador

El bucle `for` se usa cuando el número de repeticiones es conocido.

📌 Sintaxis:

```
for (inicialización; condición; actualización) {
    // Código a ejecutar en cada iteración
}
```

📌 Ejemplo:

```
for (let i = 1; i <= 5; i++) {
    console.log(`Iteración ${i}`);
}
```

📌 Explicación:

- `let i = 1`: Se inicializa la variable `i` con 1.
- `i <= 5`: La condición para seguir ejecutando el bucle.
- `i++`: Se incrementa `i` en cada iteración.

📌 Salida esperada:

```
Iteración 1
Iteración 2
Iteración 3
Iteración 4
Iteración 5
```

2.2 Bucle `while`: Repetición basada en una condición lógica

El bucle `while` ejecuta el código mientras una condición sea verdadera.

📌 Sintaxis:

```
while (condición) {
    // Código a ejecutar en cada iteración
}
```

📌 Ejemplo:

```
let contador = 1;
```

```
while (contador <= 5) {

    console.log(`Contador: ${contador}`);

    contador++;

}
```

📌 Explicación:

- `let contador = 1`: Se inicializa la variable.
- `while (contador <= 5)`: El bucle se ejecuta mientras `contador` sea menor o igual a 5.
- `contador++`: Incrementa el valor de `contador` en cada iteración.

📌 Salida esperada:

```
Contador: 1

Contador: 2

Contador: 3

Contador: 4

Contador: 5
```

2.3 Bucle `do-while`: Similar a `while`, pero se ejecuta al menos una vez

El bucle `do-while` garantiza que el bloque de código se ejecute al menos una vez, incluso si la condición es falsa desde el principio.

📌 Sintaxis:

```
do {

    // Código a ejecutar al menos una vez

} while (condición);
```

📌 Ejemplo:

```
let numero = 10;
```

```
do {
    console.log(`Número actual: ${numero}`);
    numero--;
} while (numero > 5);
```

📌 Explicación:

- `do` ejecuta el código antes de verificar la condición.
- `while (numero > 5)`: Se repetirá mientras `numero` sea mayor que 5.

📌 Salida esperada:

```
Número actual: 10
Número actual: 9
Número actual: 8
Número actual: 7
Número actual: 6
```

3. Diferencias y casos de uso de cada tipo de bucle

Tipo de Bucle	Característica Principal	Caso de Uso Ideal
`for`	Controlado por un contador con un número fijo de iteraciones	Recorrer arrays, ejecutar un bloque de código un número exacto de veces
`while`	Depende de una condición lógica, puede ejecutarse 0 o más veces	Ejecución indefinida hasta que se cumpla una condición (ej. validaciones)
`do-while`	Se ejecuta al menos una vez, sin importar la condición inicial	Garantizar que un bloque de código se ejecute antes de validar la condición

4. Uso de `for` para recorrer un array de nombres e imprimirlos

📌 Ejemplo:

```
let nombres = ["Ana", "Juan", "Pedro", "María"];


for (let i = 0; i < nombres.length; i++) {
    console.log(`Nombre: ${nombres[i]}`);
}
```

📌 Explicación:

- `nombres.length`: Indica cuántos elementos tiene el array.
- Se recorre el array con `for`, accediendo a cada elemento mediante `nombres[i]`.

📌 Salida esperada:

```
Nombre: Ana
Nombre: Juan
Nombre: Pedro
Nombre: María
```

5. Uso de `while` para pedir datos al usuario hasta que ingrese una respuesta válida

📌 Ejemplo:

```
let respuesta;


while (respuesta !== "si" && respuesta !== "no") {
    respuesta = prompt("¿Quieres continuar? (si/no)").toLowerCase();
}
```

```
console.log("Gracias por tu respuesta.");
```

📌 **Explicación:**

- El bucle `while` sigue ejecutándose hasta que el usuario ingrese `"si"` o `"no"`.
- `prompt()` permite capturar la entrada del usuario.
- `toLowerCase()` convierte la respuesta a minúsculas para evitar problemas con mayúsculas.

📌 **Caso de uso:**

- Ideal para validar formularios o solicitar confirmaciones hasta que el usuario ingrese datos correctos.

6. Uso de `do-while` para garantizar la ejecución de un bloque de código antes de evaluar la condición

📌 **Ejemplo:**

```
let numero;

do {
    numero = parseInt(prompt("Ingresa un número mayor que 10:"));
} while (numero <= 10);

console.log(`Número válido ingresado: ${numero}`);
```

📌 **Explicación:**

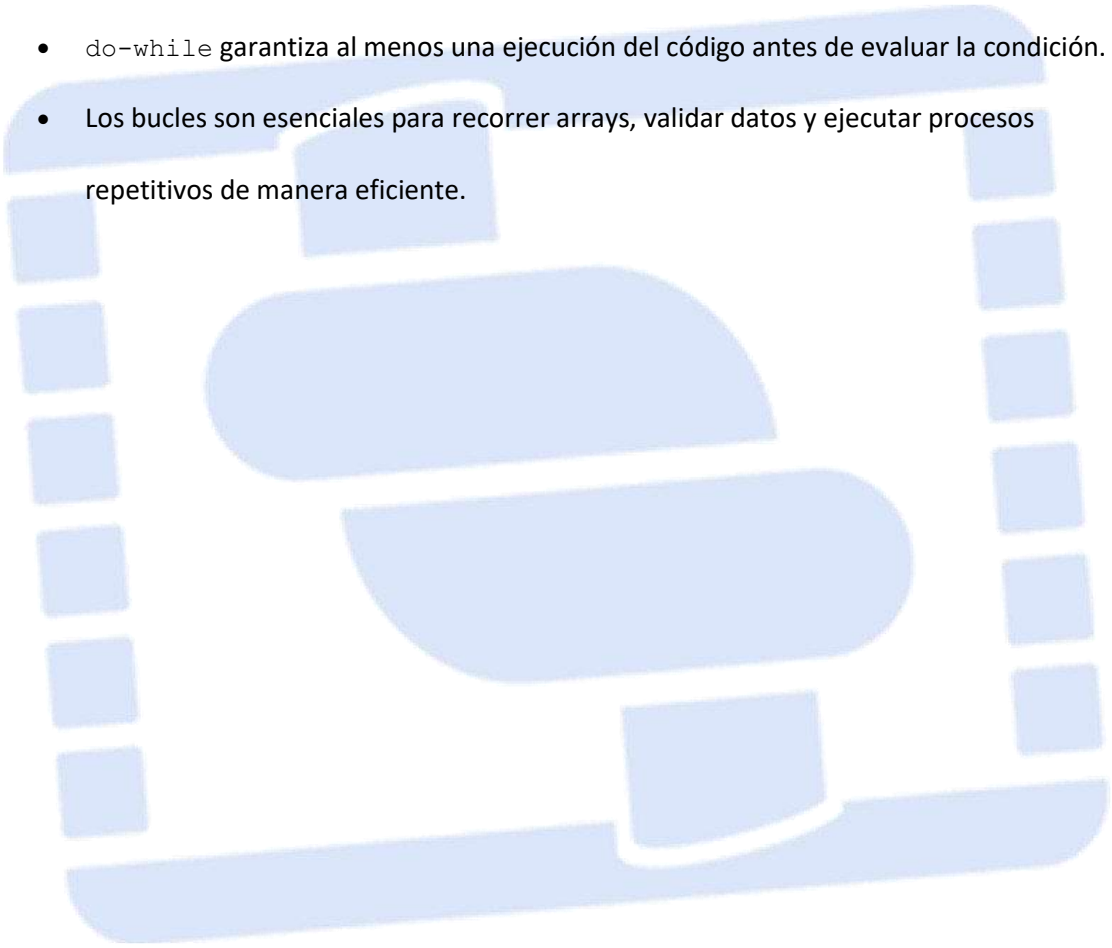
- `do` ejecuta el `prompt()` al menos una vez.
- Si el usuario ingresa un número menor o igual a 10, el bucle se repite.
- `parseInt()` convierte la entrada en número.

📌 **Caso de uso:**

- Garantizar que el usuario ingrese un dato válido antes de continuar.

Conclusión

- Los bucles permiten ejecutar un bloque de código múltiples veces según una condición.
- `for` se usa cuando se conoce la cantidad de iteraciones.
- `while` se ejecuta mientras una condición sea verdadera.
- `do-while` garantiza al menos una ejecución del código antes de evaluar la condición.
- Los bucles son esenciales para recorrer arrays, validar datos y ejecutar procesos repetitivos de manera eficiente.



INSTITUTO

KHIPU