

«Talento Tech»

Front-End JS

Clase 10



Clase 10: Estructuras de control y bucles en JavaScript

Temario:

1. Diagrama de flujo

- ¿Qué es un diagrama de flujo?
- Elementos clave de un diagrama de flujo
- Ejemplo práctico: Preparación de café
- Utilidad del diagrama de flujo en programación

2. Condicionales en JavaScript

- ¿Qué es un condicional?
- Tipos de condicionales
- Importancia de los condicionales en el desarrollo de aplicaciones
- Ejemplo práctico: Evaluación de la edad para entrar a un club

3. Operadores Lógicos y de Comparación

- ¿Qué son los operadores lógicos y de comparación?
- Comparadores.
- Operadores lógicos.
- Combinación de operadores en condiciones complejas
- Ejemplo práctico: Validación de acceso a un evento según edad y membresía

4. Bucles en JavaScript

- ¿Qué es un bucle?
- Tipos de bucles.
- Diferencias entre los tipos de bucles
- Ejemplo práctico: Iterar una lista de productos

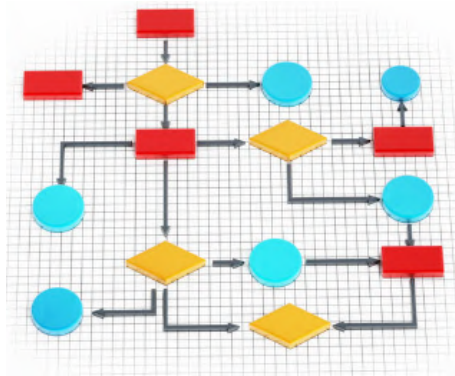
5. Cómo combinar operadores lógicos y bucles

- Combinación de operadores con ciclos
- Casos de uso en la vida real
- Ejemplo práctico: Filtrado de productos con descuentos en un catálogo

Objetivo de la clase:

En esta clase vas a aprender a planificar programas con diagramas de flujo y a usar condicionales para que tus programas tomen decisiones según distintas situaciones. Además, conocerás los bucles para repetir acciones y cómo combinarlos con condicionales para resolver problemas prácticos. Así, podrás crear código más eficiente y adaptado a diferentes necesidades.

1. Diagrama de flujo






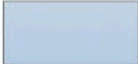

¿Qué es un diagrama de flujo?

Se trata de un mapa que muestra paso a paso cómo realizar una tarea o resolver un problema: es la representación gráfica del flujo de tu algoritmo. Es una herramienta visual súper útil, sobre todo en programación, porque te permite ver cómo se conectan las decisiones y las acciones de tu programa. Imaginá que querés preparar un café, ¿qué pasos seguirías? Un diagrama de flujo te mostraría, por ejemplo, que primero necesitás hervir el agua, luego poner el café en el filtro, y así sucesivamente.

Elementos claves de un diagrama de flujo

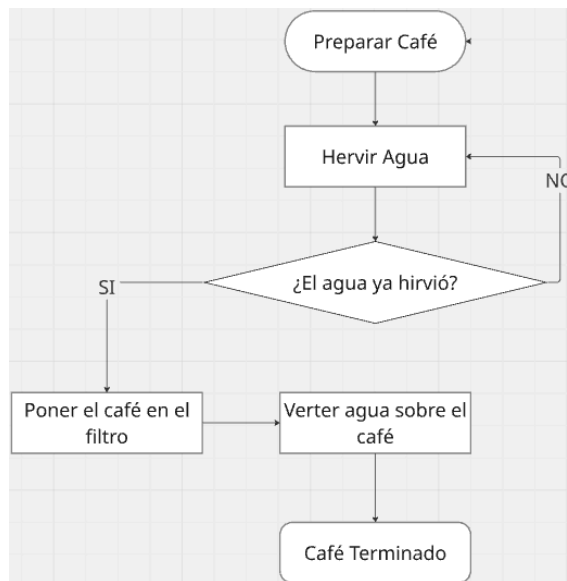
- **Inicio/Fin:** Representan el comienzo o la conclusión del proceso, se muestran en forma de óvalo.
- **Acción/Proceso:** Son las operaciones o tareas que se deben realizar, y se representan por un rectángulo.

- **Decisión:** Muestra una bifurcación en el proceso (una decisión de “sí” o “no”), se representa por un rombo.
- **Flechas:** Indican la dirección en la que fluye el proceso.

Símbolo	Nombre	Función
	Inicio / Final	Representa el inicio y el final de un proceso
	Línea de Flujo	Indica el orden de la ejecución de las operaciones. La flecha indica la siguiente instrucción.
	Entrada / Salida	Representa la lectura de datos en la entrada y la impresión de datos en la salida
	Proceso	Representa cualquier tipo de operación
	Decisión	Nos permite analizar una situación, con base en los valores verdadero y falso

Ejemplo práctico: preparar un café

1. Inicio
2. Hervir agua
3. Poner el café en el filtro
4. Verter agua sobre el café
5. Fin



Con este diagrama, antes de escribir una sola línea de código, ya tenés la idea clara de qué debe suceder y en qué orden. Usar diagramas de flujo para planificar tu código ayuda a evitar errores y mejora la organización.

2. Condicionales en JavaScript

¿Qué es una estructura condicional?

Una estructura condicional permite, en programación, la toma de decisiones. Tendrá esencialmente dos partes: **la condición** (*si...* ocurre X acción) y la acción a llevar a cabo como **consecuencia** (*entonces...* realizar Y acción). Los condicionales sólo pueden dar dos estados de respuesta, ya que la condición sólo puede ser o verdadera o falsa.

Pensémoslo en el mundo real: si salís de casa y ves que está lloviendo, vas a llevar un paraguas. En programación funciona de la misma forma: el programa evalúa una condición y decide qué acción tomar en función del resultado.

```
if (llueve) {  
  
    llevarParaguas();  
}
```



```

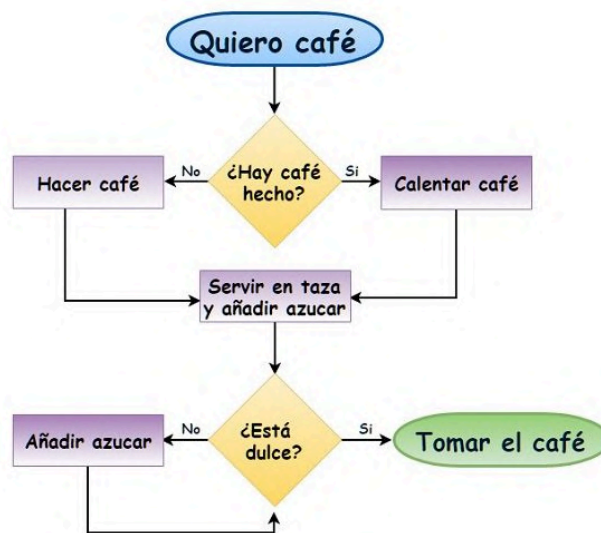
} else {

    salirSinParaguas();

}

```

Tomar un café con condicionales (rombos) sería algo como esto:



Tipos de Condicionales:

if: Este es el más básico y simplemente ejecuta un bloque de código si la condición es verdadera.

```

if (edad >= 18) {

    console.log("Sos mayor de edad.");

}

```

if...else: Esto permite ejecutar un bloque de código si la condición es verdadera, y otro en el caso de que sea falsa.

```

if (edad >= 18) {

    console.log("Sos mayor de edad.");

} else {

```

```
    console.log("Sos menor de edad.");  
  
}
```

else if: Si necesitás manejar múltiples condiciones, **else if** te da esa flexibilidad.

```
if (edad >= 18) {  
  
    console.log("Sos mayor de edad.");  
  
} else if (edad >= 13) {  
  
    console.log("Sos un adolescente.");  
  
} else {  
  
    console.log("Sos un niño/a.");  
  
}
```

Operador ternario: Es una versión compacta de **if...else**. Ideal para condiciones simples.

```
let resultado = edad >= 18 ? "mayor de edad" : "menor de edad";
```

Ejemplo práctico: evaluación de la edad

```
let edad = 20;  
  
if (edad >= 18) {  
  
    console.log("Sos mayor de edad.");  
  
} else {  
  
    console.log("Sos menor de edad.");  
  
}
```

Con esto, podés determinar si alguien puede entrar a un boliche o no, según su edad.

3. Operadores lógicos y de comparación

¿Qué son los operadores lógicos y de comparación?

Son herramientas que usás para comparar valores y tomar decisiones basadas en esas comparaciones. Por ejemplo, podés comparar si dos números son iguales, o si un valor es mayor que otro.

Comparadores

- **==** Verifica si dos valores son iguales, pero no necesariamente del mismo tipo de dato.
- **===** Compara tanto el valor como el tipo de dato.
- **!=** Verifica si dos valores son diferentes.
- **!==** Verifica si dos valores son diferentes, incluyendo su tipo de datos.
- **<, >, <=, >=** Comparan valores numéricos.

Operadores lógicos

- **&& (AND)**: Se cumplen todas las condiciones.
- **|| (OR)**: Se cumple al menos una de las condiciones.
- **! (NOT)**: Invierte el valor de la condición.

Ejemplo práctico: acceso a un evento

```
let edad = 22;

let miembroVIP = true;

if (edad >= 18 && miembroVIP) {

  console.log("Acceso concedido al área VIP.");

} else {

  console.log("Acceso denegado.");

}
```


En este ejemplo, el acceso depende tanto de la edad como de ser miembro VIP.

4. Bucles en JavaScript

¿Qué es un bucle?

Un bucle es una estructura que repite acciones en tu programa hasta que una condición deje de cumplirse. Imaginate que tenés que iterar sobre una lista de productos: con un bucle, podés hacerlo fácilmente.

Tipos de bucles

while: El ciclo se ejecuta mientras la condición sea verdadera.

```
let i = 0;

while (i < 5) {

  console.log(i);

  i++;

}
```

do...while: Similar a **while**, pero siempre ejecuta el código al menos una vez.

```
let i = 0;

do {

  console.log(i);

  i++;

} while (i < 5);
```

for: Es más compacto y se usa cuando conocés de antemano cuántas veces se repetirá el bucle.

```
for (let i = 0; i < 5; i++) {
```

```
    console.log(i);  
  }  
}
```

Ejemplo práctico: iterar productos

```
let productos = ['Laptop', 'Celular', 'Tablet'];  
  
for (let i = 0; i < productos.length; i++) {  
  
    console.log(productos[i]);  
  
}
```

Esto es un **bucle for** con tres partes:

1. `let i = 0` → la variable `i` empieza en 0 (la posición inicial del array)
2. `i < productos.length` → sigue mientras `i` sea menor que la longitud del array (productos) (en este caso 3)
3. `i++` → aumenta de a 1 cada vuelta

5. Cómo combinar operadores lógicos y bucles

En muchos casos, vas a necesitar usar operadores lógicos dentro de bucles. Por ejemplo, podés querer filtrar una lista de productos para mostrar solo los que están en descuento.

Ejemplo práctico: filtrar productos en descuento

```
let productos = [  
  
    { nombre: 'Laptop', descuento: true },  
  
    { nombre: 'Celular', descuento: false },  
  
    { nombre: 'Tablet', descuento: true }  
]
```

```
];

for (let i = 0; i < productos.length; i++) {

  if (productos[i].descuento) {

    console.log(productos[i].nombre + " tiene descuento.");}}}
```

🔴 Los corchetes **definen un array** (una lista ordenada de elementos).

Por ejemplo: `let productos = [...]`

significa que `productos` es un array.

Dentro de ese array, cada elemento se escribe también entre llaves `{ }`, porque son **objetos**.

Por ejemplo:

```
{ nombre: 'Laptop', descuento: true }
```

Esto define un objeto con clave/valor:

Con esta clase, ya dominás el uso de condicionales y bucles en JavaScript, herramientas fundamentales para cualquier aplicación dinámica. A medida que vayas practicando, estos conceptos se volverán cada vez más naturales y fáciles de implementar. ¡Adelante y a seguir experimentando!



¡Evaluación de condicionales y operadores lógicos!



Tomás (Desarrollador Senior)



¡Vamos equipo! Ahora que entendimos cómo funcionan los condicionales, operadores lógicos y bucles, es momento de ponerlos a prueba en situaciones cercanas al mundo real.

Ejercicio práctico #1:

Validación de campos de un formulario (simulado con variables)

Lucía (Product Owner)



Queremos asegurarnos de que ningún usuario envíe un formulario incompleto. Para eso vamos a simular los campos de un formulario con variables en JavaScript.

Lo que harás:

- Declarar tres variables: `nombre`, `correo` y `mensaje`, asignándoles valores de prueba.
- Crear una función que valide que ninguna de estas variables esté vacía.
- Si todos los campos tienen texto, mostrar por consola:
`"Formulario completo. Listo para enviar."`
- Si falta completar alguno, mostrar:
`"Faltan completar campos obligatorios."`

Tips de Tomás:



Usá el operador `&&` para verificar que todas las variables contengan texto. Jugá cambiando los valores de prueba para ver cómo responde tu validación.

🔴 **¡Importante!** Esta práctica te va a servir para entender la validación de datos antes de enviarlos a un servidor, algo fundamental en cualquier aplicación web.

Ejercicio práctico #2:

Iterar una lista de productos y mostrarlos con alert()

Lucía (Product Owner)



Ahora necesitamos mostrar un listado de productos para que el usuario pueda conocer la oferta disponible.

Lo que harás:

- Crear un array de al menos 5 productos (por ejemplo: "Remera", "Pantalón", "Gorra").
- Recorrer el array con un bucle `for` o `for...of`.
- En cada iteración, mostrar el producto con un `alert()`.
- Al finalizar, enviar por consola el mensaje:
"Lista de productos mostrada correctamente."



Tips de Tomás:

Definí el array como `const` para practicar buenas prácticas de variables constantes.

Podés sumar un contador para contar cuántos productos se muestran.

🔴 **¡Recordá!** Mostrar la información de forma dinámica con bucles es la base para crear interfaces interactivas más adelante.

🔴 `const` significa que **no podés volver a asignar otro array a esa misma variable**, pero **sí podés modificar el contenido** del array.

- Podés agregar, quitar o modificar elementos *dentro* del array declarado con `const`.
- Pero no podés decirle después que sea un array completamente nuevo.

Recordemos:

- `const` = su referencia no cambia.

- `let` = variable flexible, su valor puede cambiar.
- `var` = forma más vieja, hoy se recomienda evitarla.

Buenos Aires
aprende
Agencia de Habilidades para el Futuro

