

## EVALUACIÓN DE CONDICIONES LÓGICAS

#### **HINTS**

Encadenando un ternario:

0

Como se puede notar, esto conforma otra manera más de simplificar nuestro código, manteniéndolo "limpio", y a la vez, conservando su funcionalidad.

#### **BUENAS PRÁCTICAS DE CODIFICACIÓN**

De acuerdo con **The World Wide Web Consortium**, la Organización Internacional de Estándares para la **World Wide Web**, algunas de las mejores prácticas de programación al usar JavaScript, son las siguientes:

- 1. Haz nombres de variables y funciones fáciles, breves y legibles.
- 2. Evita las variables globales.
- 3. Comenta tanto como sea necesario, pero no más.
- 4. Evita mezclar con otras tecnologías.
- 5. Usa una notación abreviada cuando tenga sentido.
- 6. Modularizar: una función por tarea.
- 7. Evita mucho anidamiento.
- 8. Optimiza bucles lo más posible.



## EVALUACIÓN DE CONDICIONES LÓGICAS

- 9. Agrega funcionalidad con JavaScript, no lo uses para crear demasiado contenido.
- 10. Construye sobre buenas ideas.

0

Considera que las "mejores prácticas", se refieren a lo óptimo que los desarrolladores pueden seguir al desarrollar código.

¿Por qué no establecer como objetivo implementar, al menos una de estas buenas prácticas, en tu flujo de trabajo? En el transcurso del proceso, puedes ir aplicando más de ellas para mejorar tu trabajo como desarrollador, y la calidad del código.

#### IMPLEMENTANDO CÓDIGO A PARTIR DE UN DIAGRAMA DE FLUJO

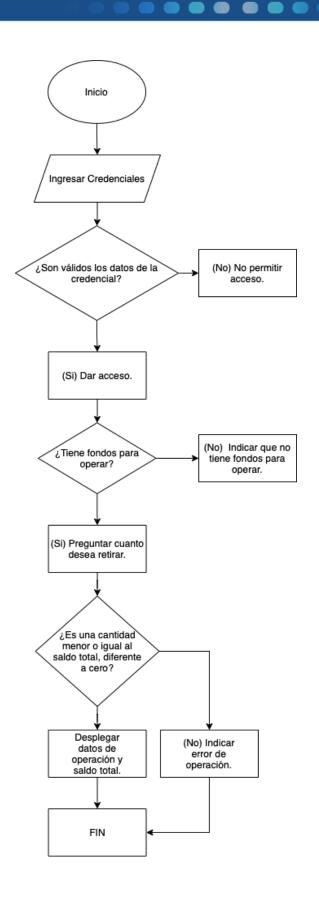
Durante este CUE se han analizado ejemplos sencillos, que permiten aprender los fundamentos del Control de Flujo, evaluando condiciones. Si se requiere abordar problemas más grandes y mucho más complejos, que necesiten condiciones de evaluación, siempre se puede considerar organizar nuestro proceso de pensamiento en un diagrama de flujo.

De hecho, éstos son una forma gráfica muy simple de representar todos los procesos necesarios para una solución informativa a un problema. Por ejemplo, si quisiéramos programar un cajero automático, se pudiese plantear en un modelo así:



0

## EVALUACIÓN DE CONDICIONES LÓGICAS





### EVALUACIÓN DE CONDICIONES LÓGICAS

Analizando con atención este diagrama, podremos determinar cómo implementar cada proceso por sí mismo. Ahora, un ejemplo de cómo transformarlo en código, es el siguiente:

```
3 let pin = 12\overline{34}
 6 let monto = 1000
 8 function ingresarCuenta(usuario, pin) {
 9
10
       cuenta usuario = (usuario === "alex123" && pin === 1234) ?
11
12
   'Bienvenido.' : 'Datos incorectos, intenta nuevamente.';
13
      console.log(cuenta usuario)
14
15
16
      transferir = prompt("desea realizar una transferencia? [si/no]");
17
18
19
      saldo = (monto > 0) ? transferir : "No tiene fondos para operar.";
20
21
22
      if (transferir === "si") {
23
24
25
           cuanto = prompt("Ingrese el monto que desea ingresar, no puede ser
26
  más que $" + monto);
27
28
29
           if (cuanto <= monto && cuanto !== 0) {
30
31
               console.log("Monto solicitado: $" + cuanto);
32
               console.log("Saldo total: $" + (monto - cuanto));
33
34
35
               console.log("No puede solicitar ese monto.");
36
37
38
39
           console.log("ok, cerrando cuenta.");
40
41
```



# EVALUACIÓN DE CONDICIONES LÓGICAS

43 // Llamamos a la funcion, iniciamos cajero. 44 ingresarCuenta(usuario, pin);

0

Se recomienda copiar el código, e interactuar con él, para revisar si hace lo que propone el diagrama de flujo. Si no, puedes intentar crear una propia implementación de esta solución, y así mejorarás tus habilidades para diseñar programas.