Logotipo

El contenido generado por IA puede ser incorrecto.

Santiago Yosa González

**Ingeniero Sistemas de Información**



Logotipo

El contenido generado por IA puede ser incorrecto.}

**Condicionales en JavaScript**

* if
* else
* else if
* switch.

Introducción Concepto de callstacky.

**Unidad I**

**Generalidades de Javascript**

**Situación**

En la sesión anterior, aprendimos a hacer comparaciones. Escribimos let edad = 20; y luego console.log(edad > 18); y la consola nos devolvió true. Pero, ¿qué hacemos con ese true? Es solo información.

Imagina que estás programando un videojuego. El jugador presiona la tecla de salto. Tu código debe **decidir**:

1. **SI (if)** el personaje está en el suelo (personaje.estaEnSuelo === true), **ENTONCES** ejecutas la animación de salto y aplicas la física para moverlo hacia arriba.
2. **SI NO (else)**, el personaje está en el aire, **ENTONCES** no haces nada (o reproduces un sonido de "ya estás saltando").

Esa capacidad de **tomar una decisión** basada en un valor booleano (true o false) es lo que separa a una calculadora de una aplicación interactiva. Los condicionales son el "cerebro" de nuestro programa, el mecanismo que nos permite ejecutar diferentes bloques de código según la situación.

**El Flujo de Ejecución y el Call Stack**

Antes de romper el flujo con condicionales, debemos entender cómo lee JavaScript.

JavaScript es **Single Threaded** (Un solo hilo). Significa que solo puede hacer una cosa a la vez. No puede sumar y restar al mismo tiempo.

**El Call Stack (Pila de Llamadas):** Imagina una pila de platos sucios.

* 1. Pones un plato encima de otro.
  2. Para lavar, debes tomar el de **arriba** (el último que pusiste) antes de llegar al de **abajo** (el primero).
  3. Esto se llama **LIFO** (Last In, First Out).

**En la práctica (por ahora):** Aunque aún no vemos funciones complejas, debes saber que el motor de JS lee tu archivo de arriba a abajo. Cuando entra en un if, "apila" esa tarea, la resuelve, y si no se cumple, la "desapila" y pasa a la siguiente línea.

**Repaso Esencial y Conexión**

Recordemos los operadores. Aprendimos que una expresión como (edad >= 18) && (tieneTicket === true) se *evalúa* y se reduce a un único valor: true o false. Hoy, usaremos esos valores booleanos como la "llave" para abrir (o mantener cerradas) puertas en nuestro código.

**Condicional Simple (if)**

Es la estructura de decisión más básica. Ejecuta un bloque de código **solo si** la condición es verdadera.

* Piensa en el if como una puerta con una cerradura.
* La condicion es la llave.
* **SI (if)** la llave es la correcta (true), la puerta se abre y puedes acceder a lo que hay dentro del bloque { ... }.
* Si la llave es incorrecta (false), la puerta permanece cerrada y el código dentro del bloque se ignora por completo.

**Ejemplo:**

// La condición (debe evaluarse a true o false)

if (condicionBooleana) {

// ... bloque de código ...

// Esto solo se ejecuta si la condición es true

}

// El código continúa aquí...

**Condicional Doble (if...else)**

Esta estructura nos da un camino alternativo. Ejecuta un bloque si la condición es true y un bloque diferente si es false.

Estás frente a una bifurcación con un letrero (la condicion).

* **SI (if)** el letrero dice "Norte" (true), tomas el camino A (el bloque if).
* **SI NO (else)**, tomas el camino B (el bloque else).
* No puedes tomar ambos caminos, y no puedes quedarte quieto. Estás obligado a elegir uno.

if (condicionBooleana) {

// ... bloque A ...

// Se ejecuta si la condición es true

} else {

// ... bloque B ...

// Se ejecuta si la condición es false

}

**Condicional Múltiple (if...else if...else)**

Nos permite encadenar múltiples preguntas en orden. Es crucial para manejar más de dos posibles resultados.

Un profesor califica un examen:

1. **SI (if)** la nota es > 90, es una "A".
2. **SI NO, PERO SI (else if)** la nota es > 80, es una "B".
3. **SI NO, PERO SI (else if)** la nota es > 70, es una "C".
4. **SI NO (else)** (para todos los demás casos), es una "F".

* **Importante:** El código comprueba las condiciones en orden y se detiene en la **primera** que sea true. Si la nota es 95, entra en el primer if y NUNCA revisará las otras condiciones.

if (condicion1) {

// Bloque A

} else if (condicion2) {

// Bloque B (Solo se revisa si la condicion1 fue false)

} else if (condicion3) {

// Bloque C (Solo se revisa si 1 y 2 fueron false)

} else {

// Bloque D (El "default", si ninguna fue true)

}

**Valores "Truthy" y "Falsy"**

Este es un concepto fundamental en JavaScript. Las estructuras if no solo comprueban true o false. Comprueban la **"veracidad"** de un valor.

En JavaScript, casi todo es "verdadero" (truthy), "Hola", 1, -10, [] (Array vacío), {} (Objeto vacío).

excepto por estos **7 valores Falsy**:

1. false
2. 0 (el número cero)
3. -0 (cero negativo)
4. "" (un string vacío)
5. null
6. undefined
7. NaN (Not a Number)

**¿Por qué es esto importante?** Te permite escribir código mucho más conciso y profesional para verificar si una variable "tiene algo".

**Anatomía del Código**

let nombreUsuario = ""; // El usuario no escribió nada

// Forma Larga (Principiante)

if (nombreUsuario !== "") {

console.log(`Bienvenido, ${nombreUsuario}`);

} else {

console.log("Bienvenido, Invitado.");

}

// Forma Corta (Profesional) - "Si nombreUsuario es truthy..."

if (nombreUsuario) {

console.log(`Bienvenido, ${nombreUsuario}`);

} else {

console.log("Bienvenido, Invitado.");

}

// Ambos códigos hacen lo mismo, pero el segundo es más idiomático de JavaScript.

**El Operador switch**

El switch es una alternativa al if...else if...else que es más limpia **solo** cuando comparas una **única variable** contra múltiples **valores exactos**.

* El switch evalúa tu variable (la opción que elegiste).
* Busca un case (caso) que coincida exactamente.
* case "Girar Dinero": -> Ejecuta el código para girar dinero.
* **Clave:** Usa comparación estricta (===).
* **break;:** Es **crucial**. Es como presionar "Finalizar". Si lo olvidas, la máquina "cae" (fall-through) y ejecuta también la siguiente opción (ej. "Consultar Saldo").
* **default::** Es la opción para "Otro" o si presionas una tecla inválida.

**Anatomía del Código**

let diaSemana = "Martes";

switch (diaSemana) {

case "Lunes":

console.log("Inicio de semana laboral.");

break; // ¡No olvides el break!

case "Martes":

console.log("Día de reunión de equipo.");

break;

case "Viernes":

console.log("¡Casi fin de semana!");

break;

default:

console.log("Es otro día de la semana.");

}

**El Operador Ternario (? :)**

Es un if...else en una sola línea. No es un *bloque* de código, sino una *expresión* (es decir, devuelve un valor).

**Analogía:**

Es un interruptor simple. Solo tiene dos estados y devuelve uno u otro.

* (condición): El interruptor.
* ?: ¿Está encendido?
* valorSiTrue: Devuelve "luz".
* :: Si no...
* valorSiFalse: Devuelve "oscuridad".

**Anatomía del Código**

let edad = 20;

// Forma if...else (Verborrágica)

let mensaje;

if (edad >= 18) {

mensaje = "Es mayor de edad";

} else {

mensaje = "Es menor de edad";

}

console.log(mensaje);

// Forma Ternaria (Concisa y Moderna)

const mensajeTernario = (edad >= 18) ? "Es mayor de edad" : "Es menor de edad";

// (condición) ? (valor si true) : (valor si false)

console.log(mensajeTernario);

**Preguntas**

**¿Cuándo uso un if...else if y cuándo un switch?**

Usa if...else if cuando necesites evaluar **rangos** o **condiciones complejas** (ej. if (edad > 18 && pais === "Colombia")). Usa switch cuando estés comprobando el valor de **una sola variable** contra una lista de **valores exactos** (ej. switch (diaDeLaSemana) { case "Lunes": ... }).

**¿Por qué if (0) es falso? Creí que if solo revisaba true o false.**

Esta es la diferencia clave entre JavaScript y otros lenguajes. El if no comprueba si la condición es *igual* a true, comprueba si es *truthy*. Como 0 (el número cero) está en la lista de los 7 valores *falsy*, el bloque if lo trata como si fuera false y no se ejecuta. Lo mismo pasa con un string vacío (""), null o undefined.

**¿Puedo poner un if dentro de otro if?**

Sí, se llama **anidamiento** (Nesting). Sin embargo, evita anidar más de 2 o 3 niveles, porque el código se vuelve difícil de leer ("Spaghetti Code"). Si necesitas anidar mucho, probablemente necesites repensar tu lógica o usar operadores lógicos &&.

**Ejemplo 1: Verificador de Contraseña**

let password = "123";

if (password.length < 8) {

console.log("Error: La contraseña es demasiado corta (mínimo 8 caracteres).");

} else {

console.log("Contraseña aceptada.");

}

**Ejemplo 2: Mensaje de Bienvenida (Ternario)**

let estaLogueado = true;

const saludo = estaLogueado ? "Bienvenido de vuelta" : "Por favor, inicia sesión"; console.log(saludo);

**Ejemplo 3: Asignación de Roles (switch)**

let codigoUsuario = "USER\_01";

let permisos;

switch (codigoUsuario) {

case "ADMIN\_00":

case "ADMIN\_01": // Se pueden apilar 'cases'

permisos = "Acceso Total";

break;

case "USER\_01":

permisos = "Acceso de Lector";

break;

default:

permisos = "Sin Acceso";

}

console.log(`El usuario tiene permisos de: ${permisos}`);

**Práctica Acompañada**

**Práctica A"Clasificador de Entradas de Cine"**

**Misión:** Determinar el precio de una entrada de cine basado en la edad del espectador.

* Niños (menores de 12): $5
* Adolescentes (12 a 17): $8
* Adultos (18 a 64): $12
* Adulto Mayor (65 o más): $6

**Paso a paso para el estudiante:**

1. **Abre tu consola.**
2. **Crea la variable de entrada:** const edad = 25;
3. **Crea la variable de salida:** let precioEntrada;
4. **Escribe el if principal:** Comprueba si edad < 12.
5. **Encadena los else if:** Añade las condiciones para los otros rangos de edad (edad <= 17, edad <= 64).
6. **Añade el else final:** El else cubrirá a los adultos mayores.
7. **Muestra el resultado:** Imprime el precio en consola.

**Solución**

const edad = 25;

let precioEntrada;

if (edad < 12) {

precioEntrada = 5;

} else if (edad <= 17) {

// Solo llega aquí si es 12 o más.

precioEntrada = 8;

} else if (edad <= 64) {

// Solo llega aquí si es 18 o más.

precioEntrada = 12;

} else {

// Solo llega aquí si es 65 o más.

precioEntrada = 6;

}

console.log(`La edad es ${edad} años. El precio de la entrada es: $${precioEntrada}`);

**Troubleshooting**

**El error de asignación en el if:**

* *Error:* if (vida = 0) { ... }
* *Explicación:* Un solo igual = **asigna**. Estás forzando a que vida sea 0.
* *Solución:* Siempre usa === (triple igual) para comparar. if (vida === 0).

**Olvidar el break en el switch:**

* *Síntoma:* El código ejecuta el caso correcto, ¡pero también todos los que están debajo!
* *Solución:* Revisa que cada case termine con break;.

**Confusión de Tipos:**

* *Error:* if (edad === "18") cuando edad es un número 18.
* *Solución:* Recuerda que === revisa valor Y tipo. Convierte tus inputs o asegúrate de usar el tipo correcto.

**Ejercicio Independiente - Calculadora de Costo de Envío E-Commerce**

Vas a simular la lógica de envíos de Amazon. Debes calcular el costo de envío basándote en dos variables: destino (Nacional o Internacional) y peso (en kg).

**Reglas de Negocio:**

1. Si el destino es "Nacional":
   * Peso < 5kg: Costo $10.
   * Peso >= 5kg: Costo $20.
2. Si el destino es "Internacional":
   * Peso < 5kg: Costo $30.
   * Peso >= 5kg: Costo $50.
3. Crea las variables destino y peso.
4. Usa if anidados o lógica && para determinar el costo.
5. Imprime: "El envío a destino [Destino] con peso [Peso]kg cuesta $[Costo]".

**Criterios de logro:**

* Uso correcto de comparadores.
* Lógica anidada o compuesta funcional.
* Mensaje final dinámico con Template Literals.

**¿Debería Saber Algo Más?**

**. "Short-circuiting" (Cortocircuito) con && y ||**

Los operadores lógicos && y || no siempre devuelven true o false. Tienen un comportamiento especial:

* A || B: Evalúa A. Si A es *truthy*, devuelve A y nunca evalúa B. Si A es *falsy*, devuelve B.
* A && B: Evalúa A. Si A es *falsy*, devuelve A y nunca evalúa B. Si A es *truthy*, devuelve B.

**El Operador "Nullish Coalescing" (??) (ES2020 - Avanzado)**

Este es el || moderno y más seguro.

* El problema del || es que reacciona a *todos* los valores falsy. Si let cantidad = 0; y haces const valor = cantidad || 10;, valor será 10, lo cual es un error (0 era un valor válido).
* El ?? (Nullish Coalescing Operator) **solo** reacciona a null o undefined.

**Regla moderna:** Usa ?? para valores por defecto, a menos que *específicamente* quieras tratar 0 y "" como inválidos.

Logotipo

El contenido generado por IA puede ser incorrecto.