Operador ternario en JavaScript

El operador condicional (ternario) en JavaScript es un atajo para escribir condiciones simples en una sola línea. Funciona como un `if` breve: evalúa una condición y retorna un valor u otro según si la condición es verdadera o falsa[1][2]. La sintaxis básica es:

```
condición ? exprSiTrue : exprSiFalse
```

Por ejemplo:

```
let acceso = edad >= 18 ? "Permitido" : "Denegado";
```

Aquí, si `edad >= 18` es `true`, la variable `acceso` recibe `"Permitido"`, de lo contrario `"Denegado"`[3]. MDN explica que este operador “es el único operador en JavaScript que tiene tres operandos” y que se usa “con frecuencia como atajo para la instrucción `if`”[1]. En otras palabras, es una forma muy breve de realizar una acción basada en una condición[4][2].

Ventajas del operador ternario frente a `if/else`: El ternario permite escribir condiciones simples de forma muy concisa, evitando varias líneas de código. Esto es útil especialmente cuando quieres asignar un valor a una variable en función de una condición[3]. Por ejemplo, en lugar de escribir:

```
let mensaje;
if (isMember) {
  mensaje = "¡Bienvenido, miembro!";
} else {
  mensaje = "Por favor regístrate.";
}
```

puedes hacer lo mismo con un ternario en una sola línea:

```
let mensaje = isMember ? "¡Bienvenido, miembro!" : "Por favor regístrate.";
```

En general, el ternario enfatiza qué valor se asigna según la condición, mientras que el `if/else` enfatiza el flujo de control (primero la condición, luego las acciones). Si la lógica es muy simple (una asignación o retorno basado en una condición), el ternario puede hacer el código más limpio. Sin embargo, si necesitas ejecutar varias líneas de código o comprobaciones complejas, un bloque `if/else` tradicional suele ser más claro y legible. Por ejemplo, anidar varios ternarios puede confundir a quien lea el código. En casos con muchas condiciones o lógica más extensa, es preferible usar `if/else` para mantener el código fácil de entender.

Ejemplos prácticos con el operador ternario

- Asignación sencilla: Asignar un valor según una condición:

```
let edad = 20;
let etiqueta = (edad >= 18) ? "Mayor de edad" : "Menor de edad";
console.log(etiqueta); // "Mayor de edad"
```

- Encadenando ternarios: Aunque se puede anidar ternarios, suele desaconsejarse porque puede quedar difícil de leer. Por ejemplo:

```
let puntaje = 85;
let calificacion = (puntaje >= 90) ? "Sobresaliente"
                 : (puntaje >= 75) ? "Bien"
                 : (puntaje >= 60) ? "Suficiente"
                 : "Insuficiente";
console.log(calificacion); // "Bien"
```

Este código funciona, pero verás que es más complicado de seguir que usar varios `if/else` tradicionales. Para casos así quizá sería mejor:

```
let calificacion;
if (puntaje >= 90) {
  calificacion = "Sobresaliente";
} else if (puntaje >= 75) {
  calificacion = "Bien";
} else if (puntaje >= 60) {
  calificacion = "Suficiente";
} else {
  calificacion = "Insuficiente";
}
```

- Uso con operaciones simples: A veces se usa el ternario para ejecutar pequeñas operaciones. Por ejemplo:

```
const wasCancelled = false;
wasCancelled ? console.log("Acción cancelada") : console.log("Acción
continuada");
```

En resumen, el operador ternario es útil para escribir condiciones simples de forma breve[2]. Sin embargo, no reemplaza a `if/else` para lógicas más complicadas.

Eventos en JavaScript

En JavaScript, un evento es “algo que sucede en el sistema que estás programando” y del cual el navegador te avisa para que puedas responder con código[5]. Por ejemplo, cuando el usuario hace clic en un botón, presiona una tecla, carga o recarga una página, envía un formulario, mueve el ratón sobre un elemento, etc., todo eso dispara eventos[6][7]. El navegador ofrece muchos tipos de eventos para reaccionar a casi cualquier interacción del usuario o cambio en la página.

Para manejar (atrapar) un evento, se le asigna un manejador o listener, que es una función JavaScript que se ejecuta cuando ocurre ese evento. Hay varias formas de hacerlo, pero la más recomendada es usar el método `addEventListener`. Por ejemplo,

si quieres reaccionar a un clic en un botón, primero obtienes una referencia al botón y luego registras un escuchador para el evento `"click"`:

```
<button id="miBoton">Haz clic</button>
<script>
  const boton = document.getElementById("miBoton");
  boton.addEventListener("click", () => {
    alert("¡Se hizo clic en el botón!");
  });
</script>
```

En este código, al hacer clic en `<button>`, se ejecuta la función que muestra un `alert`. MDN señala que los objetos que pueden lanzar eventos (como elementos del DOM) tienen el método `addEventListener()`, y éste es el mecanismo recomendado para registrar manejadores de eventos[8].

También existe la forma clásica de asignar eventos usando propiedades o atributos HTML. Por ejemplo, en W3Schools se muestra que puedes hacer:

```
<button onclick="mostrarFecha()">Presiona para fecha</button>
<script>
  function mostrarFecha() {
    document.getElementById("demo").innerHTML = Date();
  }
</script>
```

o bien en JavaScript:

```
document.getElementById("miBoton").onclick = mostrarMensaje;
```

En ambos casos, la función asociada se ejecutará cuando ocurra el evento de clic[9]. Sin embargo, esta forma (`onclick=`) es menos flexible que `addEventListener`, por lo que hoy en día se recomienda usar `addEventListener`.

A continuación algunos ejemplos de eventos comunes y su manejo:

- Click en botón:

  ```
  <button id="btn">Click me</button>
  <script>
    const btn = document.getElementById("btn");
    btn.addEventListener("click", () => {
      document.body.style.backgroundColor = "lightblue";
    });
  </script>
  ```

  Cuando el usuario hace clic, el manejador cambia el color de fondo de la página.

- Teclas del teclado:

```
document.addEventListener("keydown", (event) => {
  console.log(`Tecla presionada: ${event.key}`);
});
```

Este código escucha cualquier pulsación de tecla en el documento. Cada vez que se presiona una tecla, registra en la consola cuál fue (por ejemplo, “a” o “Enter”).

- Otros ejemplos de eventos: En HTML/DOM hay eventos para casi todo: carga de página (`load`), envío de formulario (`submit`), movimiento del ratón (`mouseover`, `mouseout`), cambios en campos de texto (`input`, `change`), etc. Por ejemplo, `window.addEventListener("resize", ...)` detecta cambios de tamaño de la ventana. W3Schools enumera eventos típicos como “un clic en el ratón” o “una página web ha cargado”[6]. En cada caso, basta con registrar un listener apropiado para ejecutar la lógica deseada.

En resumen, los eventos en JavaScript permiten reaccionar a las acciones del usuario o cambios en la página. Para manejarlos normalmente usamos `addEventListener(evento, función)` sobre el elemento correspondiente[8][6]. Al ocurrir el evento (por ejemplo, un clic), el navegador llama a la función que asociamos y ahí podemos ejecutar el código necesario.

Fuentes: Documentación oficial y tutoriales como MDN Web Docs y W3Schools describen estos conceptos con detalle[1][5][2][9].

---

[1] [3] Operador condicional (ternario) - JavaScript | MDN

https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Operators/Conditional_operator

[2] JavaScript Conditional Ternary Operator

https://www.w3schools.com/js/js_if_ternary.asp

[4] JavaScript Course Conditional Operator in JavaScript - GeeksforGeeks

https://www.geeksforgeeks.org/javascript/javascript-course-conditional-operator-in-javascript/

[5] [7] [8] Introducción a los eventos - Aprende desarrollo web | MDN

https://developer.mozilla.org/es/docs/Learn_web_development/Core/Scripting/Events

[6] [9] JavaScript DOM Events

https://www.w3schools.com/js/js_htmldom_events.asp