

25¢

¡EL LENGUAJE QUE LE DIO RITMO A LA WEB!

APPROVED
BY THE
COMIC
CODE
AUGUST
1978

¡IMISIÓN: DOM DINÁMICO! ¡UNA ODISEA DIGITAL CON JAVASCRIPT!

```
document.querySelector('.hero')  
  
.addEventListener('click', () => {  
  console.log('Power up!');  
});
```



WEB
COMICS
GROUP

ISSUE #1 - OCT 1978

NotebookLM

LA FORTALEZA ESTÁTICA

Mira, Leo. Esa es la 'Fortaleza Estática'. Puro HTML. Fuerte, estructurada, pero... sin alma. Sin dinamismo.

<h1>FORTALEZA ESTÁTICA</h1>

<p>ESTE SITIO ES PURO HTML. SIN ESTILO. SIN VIDA. ESTRUCTURA BÁSICA.</p>

```
<ul>
  <li>ELEMENTO 1</li>
  <li>ELEMENTO 2</li>
  <li>ELEMENTO 3</li>
</ul>
```

```
<html><body><h1>FORTALEZA ESTÁTICA</h1><p>ESTE SITIO ES PURO HTML. SIN ESTILO. SIN VIDA. ESTRUCTURA BÁSICA.</p><ul>
  <li>ELEMENTO 1</li><li>ELEMENTO 2</li><li>ELEMENTO 3</li>
</ul></body></html>
```

¿Y cómo le damos vida, Ada?

Con la 'energía'. Con el poder de JavaScript. Es el lenguaje de programación que nos permite crear interactividad dinámica en los sitios web. Desde carruseles de imágenes hasta aplicaciones complejas.

JavaScript es un robusto lenguaje de programación que se puede aplicar a un documento HTML para crear interactividad.





String (Cadena): Es una secuencia de texto.
Siempre va entre comillas.



Number (Número): Para cualquier tipo de número, entero o decimal. ¡Sin comillas!

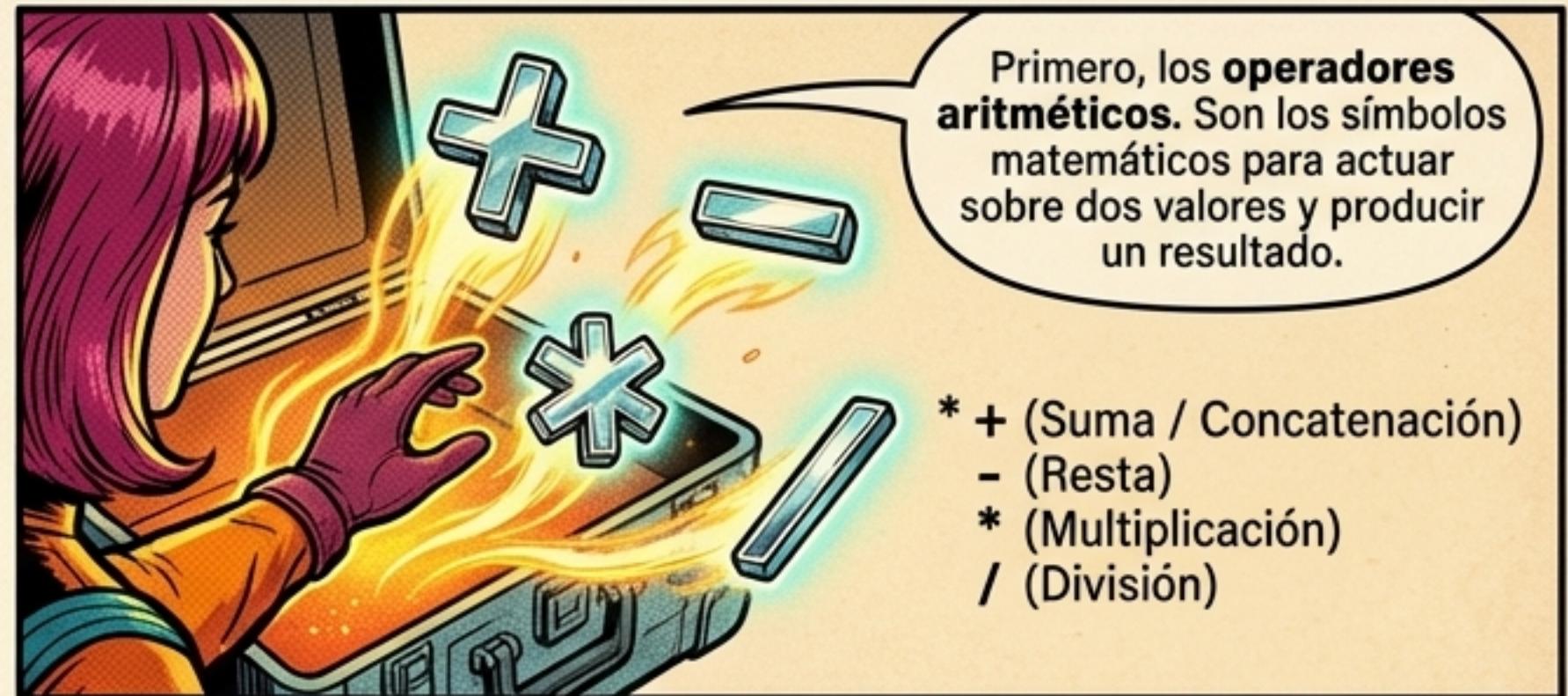


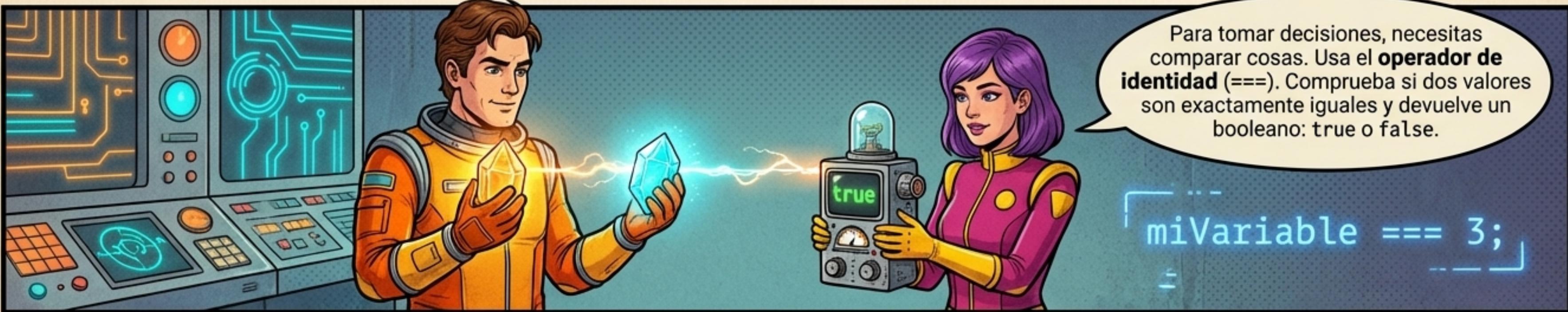
Boolean (Booleano): Solo puede ser verdadero (**true**) o falso (**false**). Es la base de todas las decisiones.

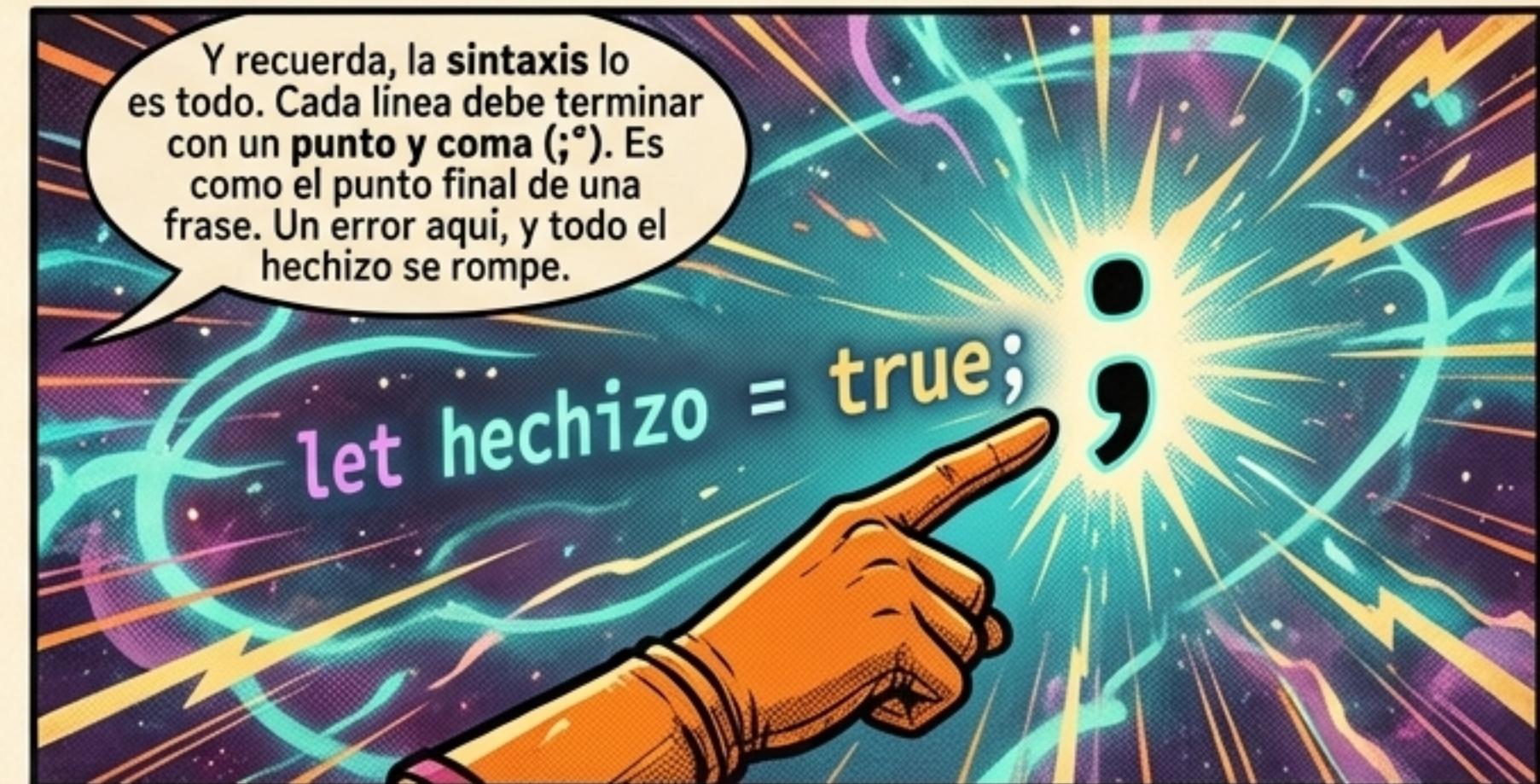


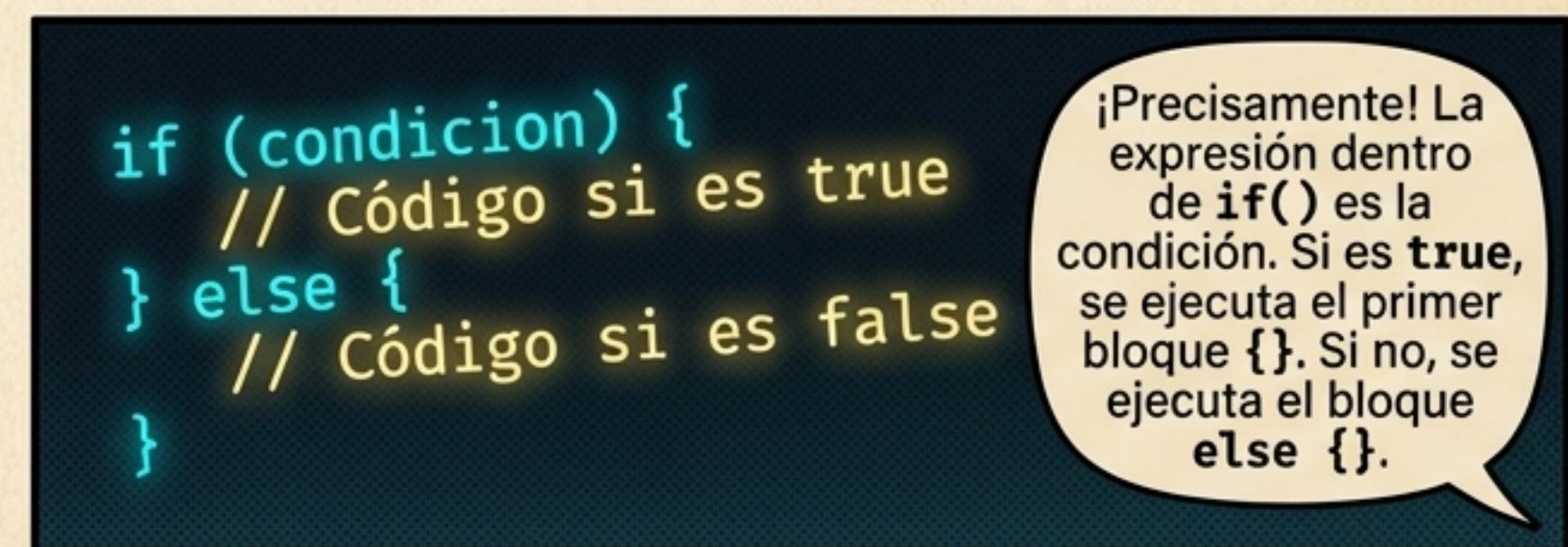
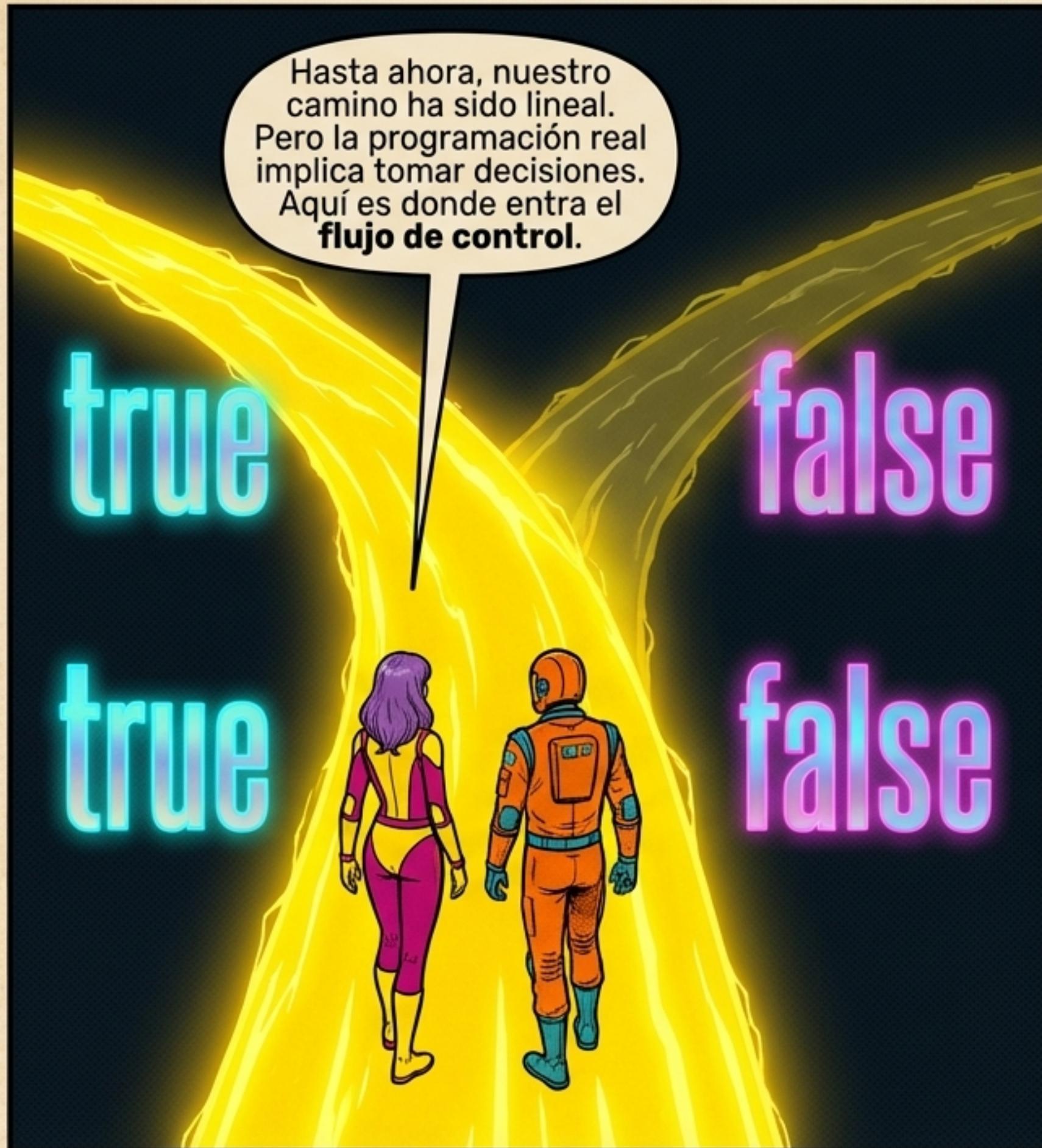
Y estos son los pesos pesados: **Array**, una estructura para almacenar múltiples valores, y **Object**, que puede guardar... ¡prácticamente de todo!











Veamos un ejemplo. Si la variable `luzEncendida` es `true` ...

```
let luzEncendida = true;  
  
if (luzEncendida === true) {  
    alert("La luz está encendida!");  
} else {  
    alert("La habitación está a oscuras...");
```

¡Entendido! Como `luzEncendida` es `true`, tomamos el primer camino y se ejecuta el primer `alert` .

true

false

¡POP!

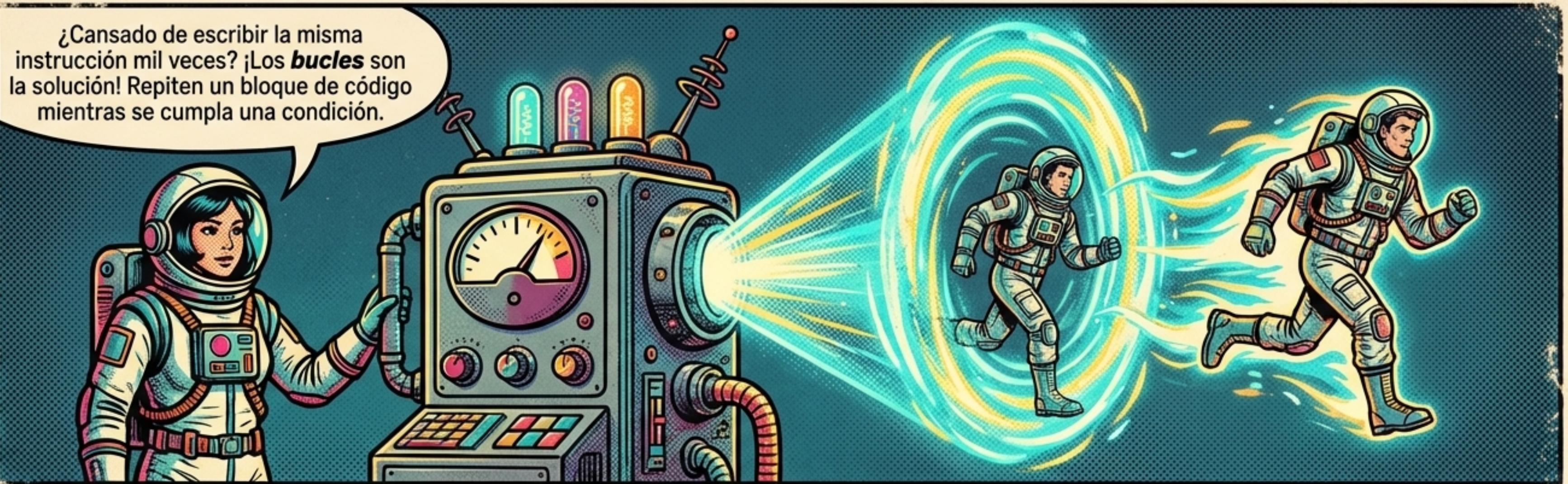
¡La luz está encendida!

¡Correcto! Así es como tus programas empiezan a reaccionar y a tomar sus propias decisiones.



switch evalúa una variable y, dependiendo de su valor, ejecuta el bloque de código del `case` correspondiente. ¡Es como una centralita de opciones!





El bucle **for** es perfecto cuando sabes cuántas veces quieres repetir algo. Tiene tres partes: inicialización, condición y actualización.

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

let i = 0: Empezamos a contar desde 0.

i < 5: Repetimos mientras i sea menor que 5.

i++: Aumentamos i en 1 en cada vuelta.

El bucle **while** se repite mientras una condición sea **true**. Es ideal cuando no sabes exactamente cuántas iteraciones necesitarás.

```
while (energia > 0) {  
    ...  
}
```

EL DOM: CORAZÓN DEL CÓDIGO

¡Bienvenido, Leo, al corazón de la máquina! ¡El Document Object Model, o como lo llamamos los colegas, el DOM!

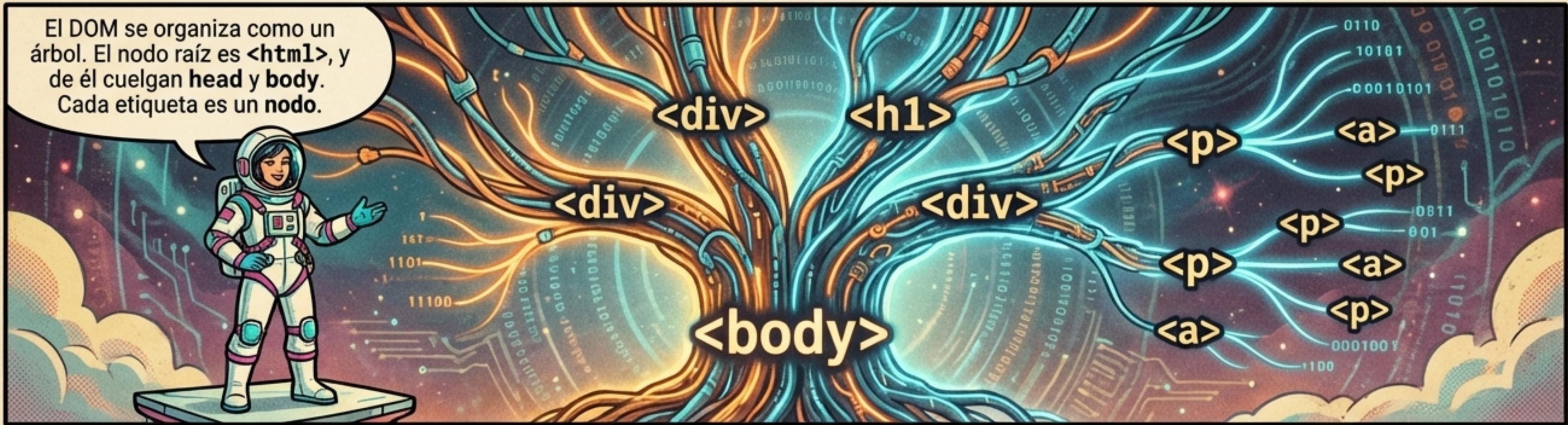
<html>

Cuando un navegador carga una página, crea una representación del documento HTML en memoria. Esa representación es el **DOM**. Es el puente que conecta JavaScript con la página web, permitiéndonos manipularla.

¿Es... es el código HTML?

Es más que eso. Es una estructura de objetos viva. Y vamos a aprender a escalarla... y a remodelarla.

El DOM se organiza como un árbol. El nodo raíz es `<html>`, y de él cuelgan **head** y **body**. Cada etiqueta es un **nodo**.







¡Bien hecho, Leo!
Ahora que lo tienes,
vamos a cambiar lo
que dice.



Si quieras
meterle más
ritmo y añadir HTML,
usa ` .innerHTML`.
Pero ¡cuidado! Es
muy potente y puede
ser peligroso si no
controlas lo que
insertas.



No solo podemos cambiar lo que existe... ¡podemos crear vida nueva! Con `document.createElement('p')` creas un nuevo nodo de párrafo.

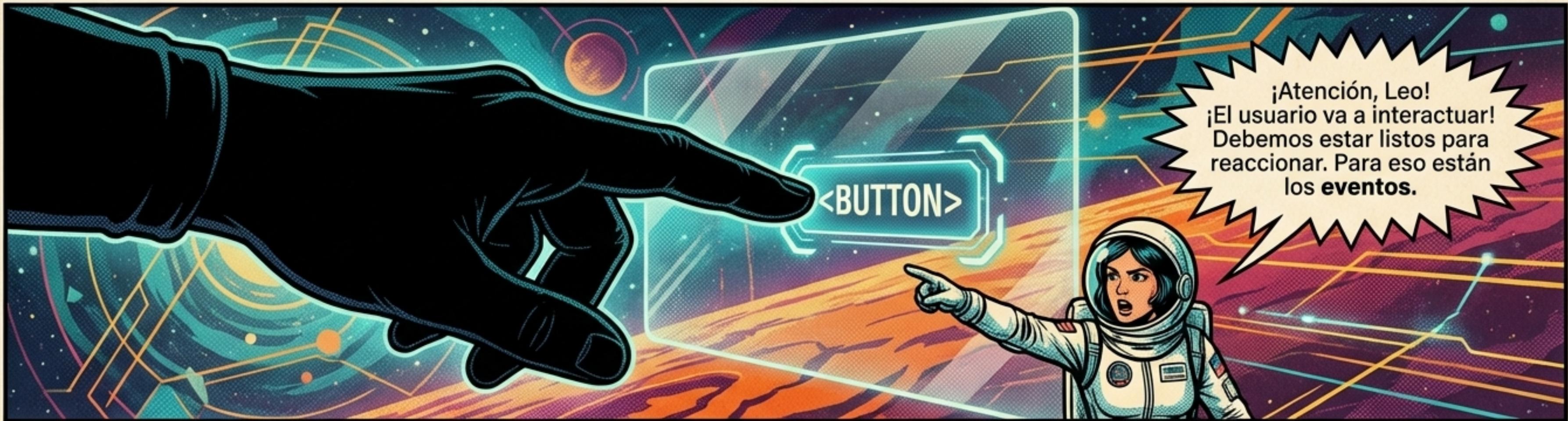


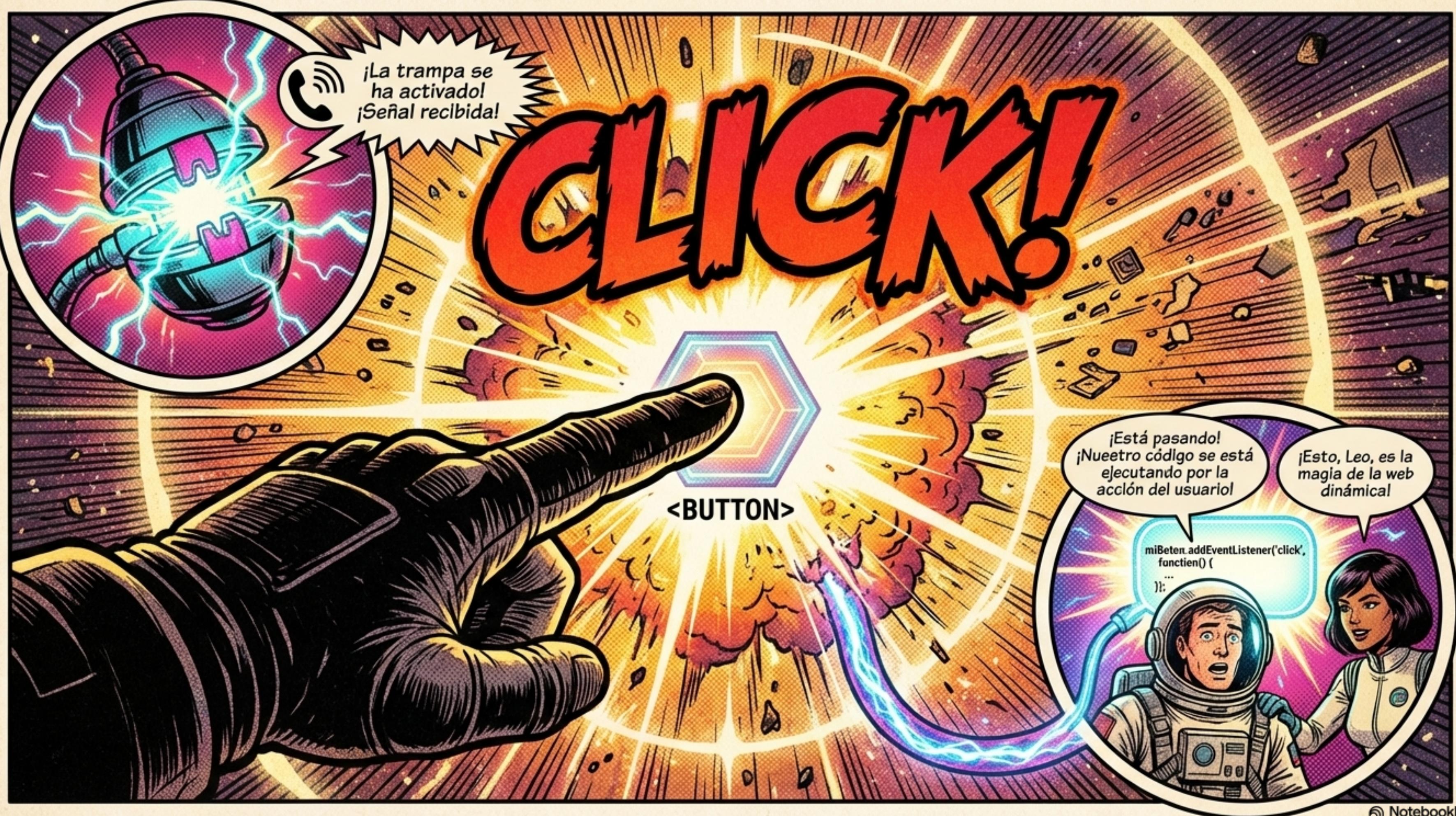
Una vez creado, debes añadirlo al DOM. Usa `appendChild()` para insertarlo como un hijo al final de otro elemento.



Y lo que se crea, se puede destruir. El método `.remove()` elimina un nodo del DOM para siempre.







PISTOLA DE ESTILOS

Último paso para darle buen rollo a este sitio: ¡color! Con esta 'Pistola de Estilos' puedes cambiar el CSS de cualquier elemento en tiempo real.



Usa la propiedad `.style` para cambiar atributos CSS específicos, como `backgroundColor` o `width`.



```
miCaja.style.backgroundColor = 'red';
```

Para cambios más radicales, usa `.classList`. Puedes añadir (`.add()`), quitar (`.remove()`) o alternar (`.toggle()`) clases CSS que ya tengas definidas. ¡Es más limpio y potente!



```
miCaja.classList.toggle('highlight');
```



CÓDIGO COMPLETO

```
const input = document.getElementById('nameInput');
const output = document.getElementById('output');
const button = document.getElementById('submitButton');

button.addEventListener('click', () => {
  const userName = input.value; // Obtener el valor
  output.textContent = `¡Qué onda, ${userName}!` // Mostrarlo
});
```

RESULTADO DINÁMICO

Nombre:
Leo

Enviar

CLICK!

CLICK!

Nombre:
Leo

<p>

¡Qué onda, Leo!

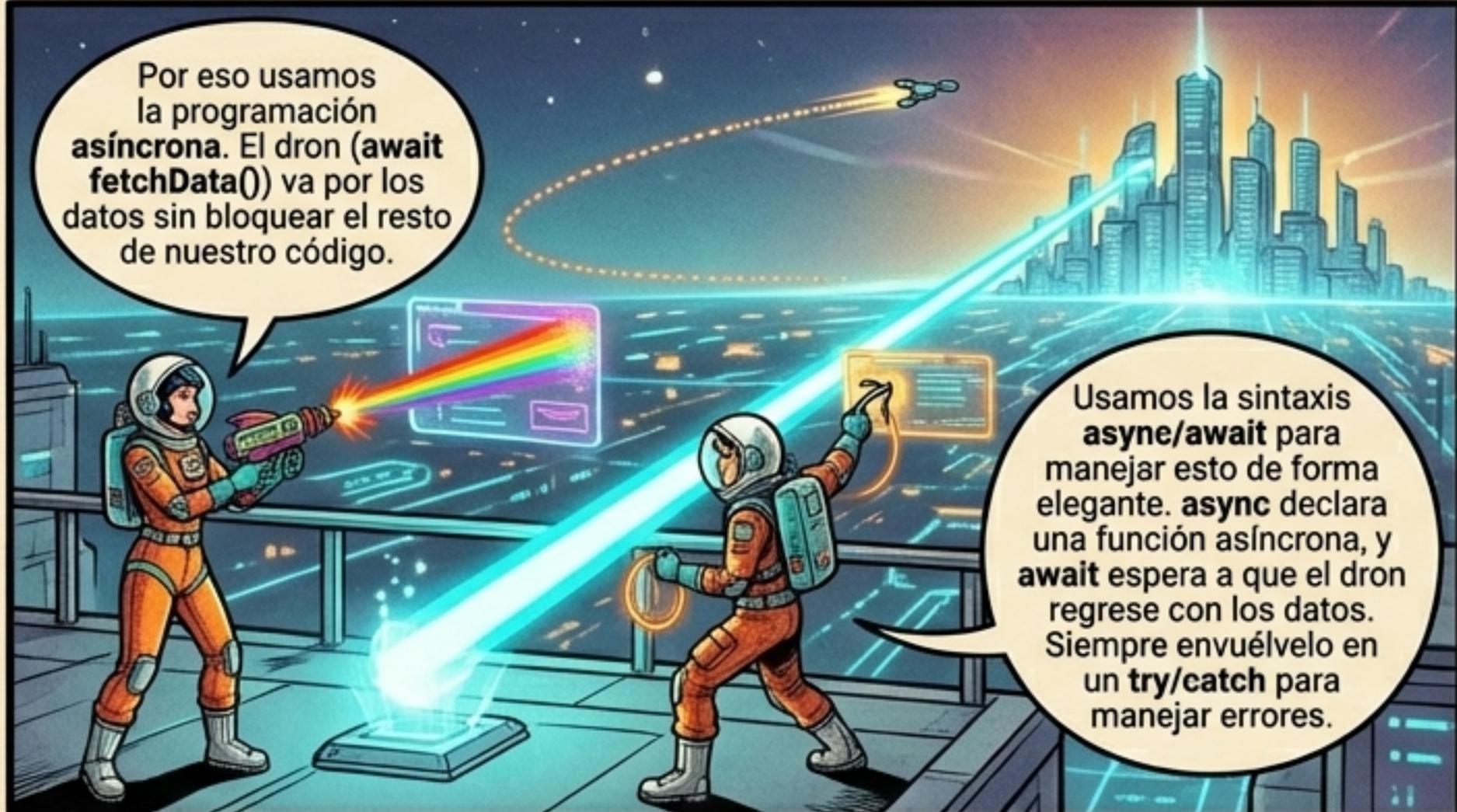
¡Perfecto!
Has capturado la
entrada del usuario y
has modificado el DOM
para reflejarla. ¡La
fortaleza ya no es
estática!

ASYNC: EL PODER DE ESPERAR SIN PARAR

Nuestra misión principal está completa, pero tu viaje apenas comienza. Hay poderes más grandes. A veces, necesitas pedir datos a un servidor lejano, una API. Si lo haces de forma normal (**síncrona**), ¡toda nuestra página se congelaría esperando!



Por eso usamos la programación **asíncrona**. El dron (`await fetchData()`) va por los datos sin bloquear el resto de nuestro código.



Usamos la sintaxis **async/await** para manejar esto de forma elegante. **async** declara una función asíncrona, y **await** espera a que el dron regrese con los datos. Siempre envuélvelo en un **try/catch** para manejar errores.

```
const getData = async () => {
  try {
    const response = await fetchData(); // El dron va por los datos
  } catch (error) { // La red de seguridad
    console.error(error);
  }
};
```

try

catch



¡SEGURIDAD Y AUTENTICACIÓN!



INTRODUCCIÓN A TYPESCRIPT: EL SIGUIENTE NIVEL!



