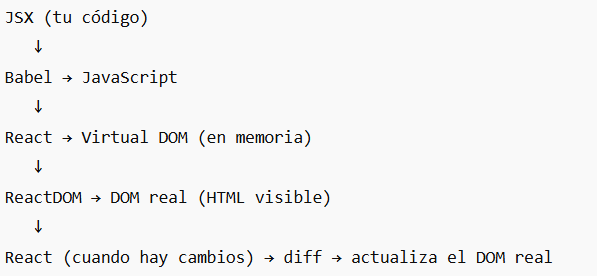
Variables

| **Palabra clave** | **Se puede reasignar** | **Se puede redeclarar** | **Alcance** | **Ejemplo** |
| --- | --- | --- | --- | --- |
| var | ✅ | ✅ | función / global | var x = 10; |
| let | ✅ | ❌ | bloque {} | let x = 10; x = 20; |
| const | ❌ | ❌ | bloque {} | const x = 10; |

Comillas

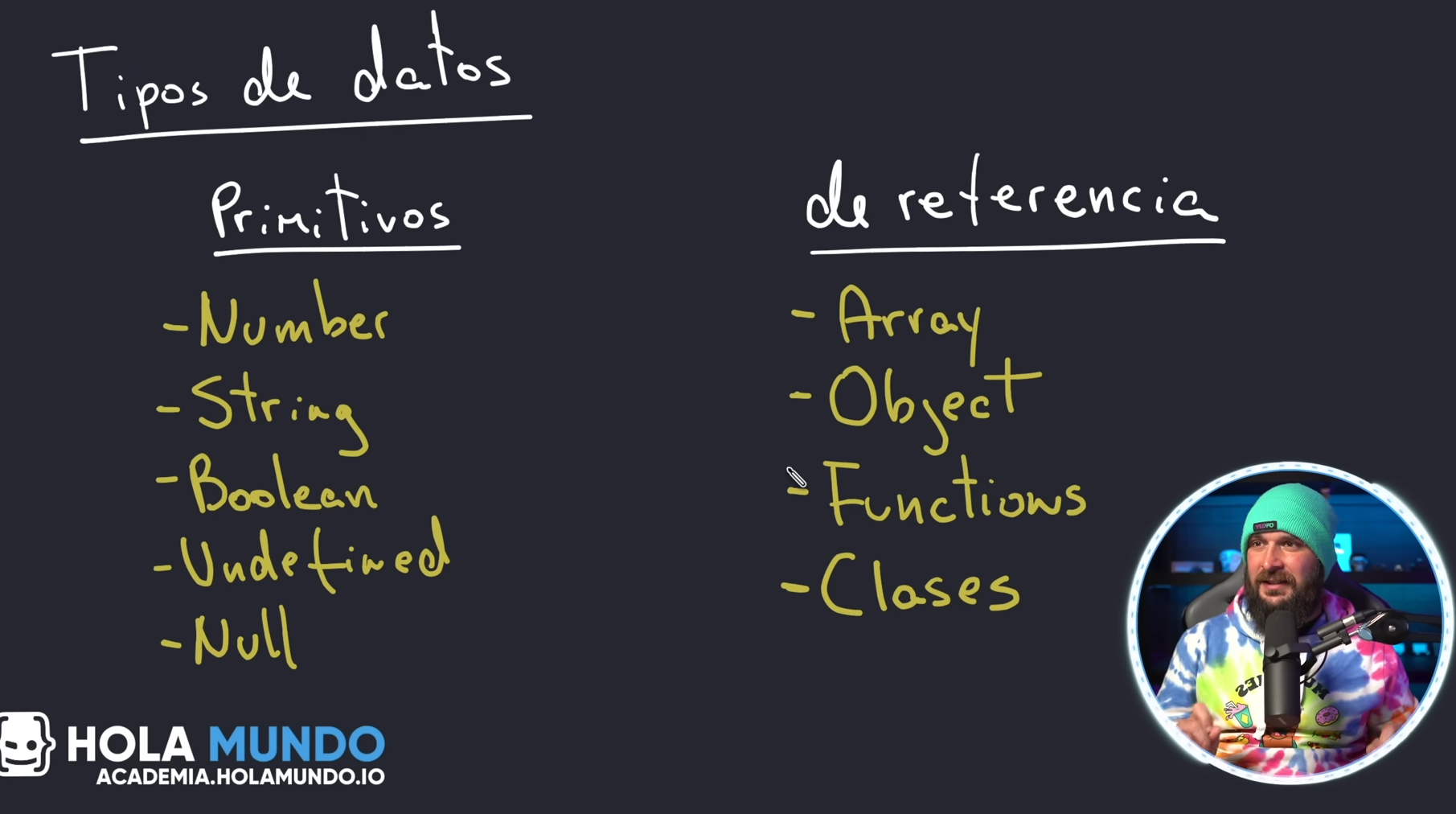
| **Lenguaje** | **" "** | **' '** | **` `** |
| --- | --- | --- | --- |
| **C#** | para strings | para chars | ❌ no existe |
| **JavaScript** | para strings | para strings (igual que " ") | para *template literals* (permite interpolación y multilínea) |

Cómo funciona React:  
JSX: Mezcla entre JS + HTML (pero no es html), React lo pasa a JS puro.

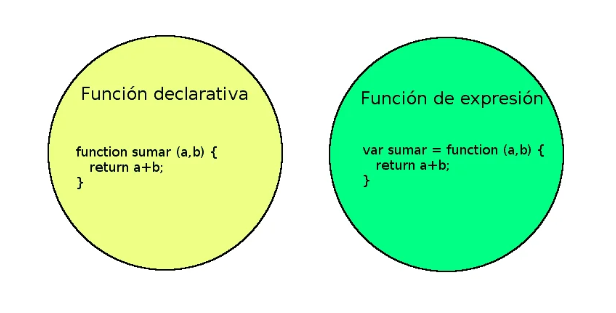


Nota: no se actualiza todo el DOM, sino el componente del Don que cambió en el DOM virtual.

| **Etapa** | **Qué ocurre** | **Dónde pasa** |
| --- | --- | --- |
| 1 Transpilación (JSX → JS) | Babel/Vite transforma el código | 💻 **En tu PC (antes de ejecutar)** |
| 2 Ejecución del JS | React se inicia | 🌐 **En el navegador** |
| 3 Creación del Virtual DOM | React crea objetos JS en memoria | 🌐 **En el navegador** |
| 4 Render del DOM real | ReactDOM pinta el HTML visible | 🌐 **En el navegador** |
| 5 Actualizaciones por estado | React hace diff y actualiza partes | 🌐 **En el navegador** |

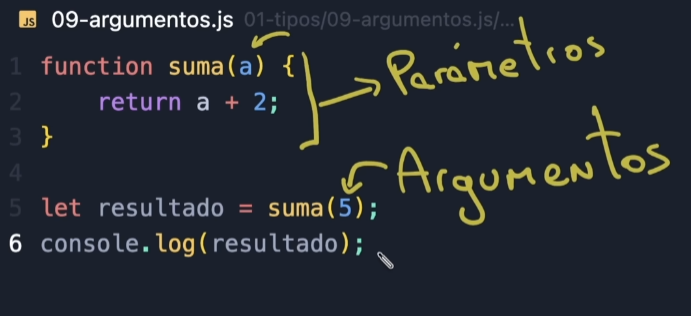


* Cuando pasás un valor **primitivo**, la función recibe **una copia**, no el valor original.  
  Cambiarlo dentro de la función **no afecta** a la variable fuera de ella.
* Cuando pasás un **objeto o arreglo**, la función recibe una **referencia** (una especie de puntero al mismo espacio de memoria).  
  Entonces, si modificás **sus propiedades internas**, el cambio sí se refleja afuera.

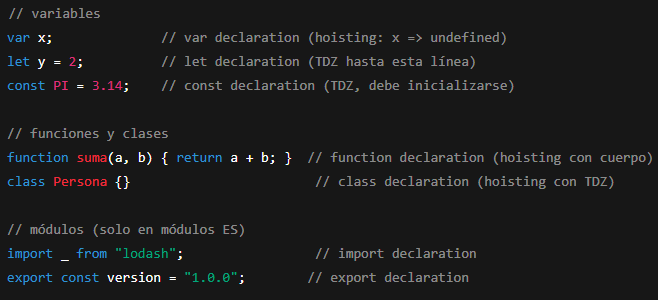


**Formas de declarar un objeto**

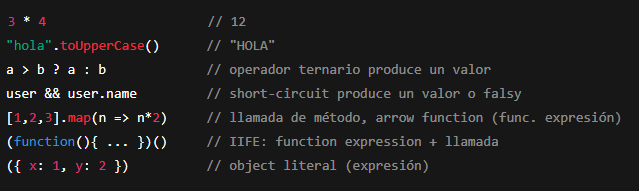
| **Método** | **Sintaxis** | **Cuándo usar** |
| --- | --- | --- |
| Literal {} | const obj = {} | Simple, más usado |
| new Object() | const obj = new Object() | Obsoleta / poco usada |
| Función constructora | function Persona(){} | JS clásico |
| Clase class | class Persona {...} | Moderno y legible |
| Object.create() | Object.create(proto) | Herencia sin clases |
| Factory Function | return { ... } | Patrón flexible y limpio |
| Object.assign() | Object.assign(dest, src) | Fusionar / clonar |
| Propiedades dinámicas | { [key]: value } | Claves calculadas |



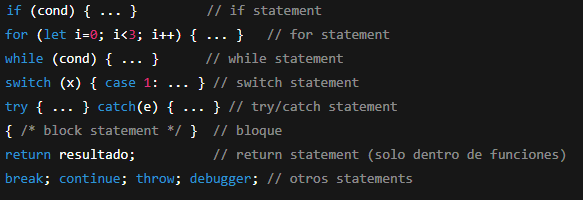
**Declaration (Declaración)** 🡪 Cuando se crea algo que vas a referenciar en un futuro. NO se le puede hacer console.log().

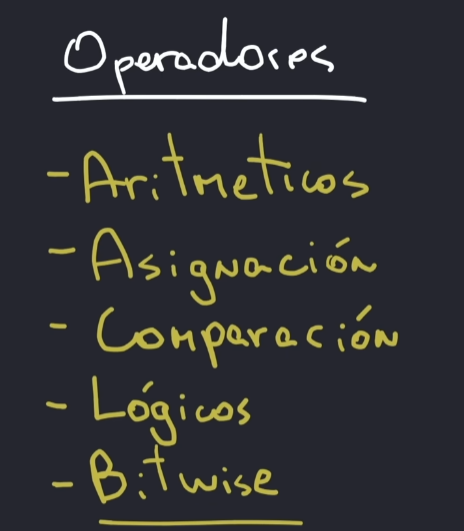


**Expression (Expresión**) 🡪 Es cualquier línea de código o porción de código que evalué en un valor. Dicho de otra manera, cada vez que el código hace algo y puedes obtener un valor de este estamos ante una expression. Se le puede hacer console.log().



**Statement (Expresión)** 🡪 Orden/estructura que el motor **ejecuta**. Controla flujo, define bloques, etc. Si te permite escribir lógica o realizar ciertas tareas estamos ante un Statement.



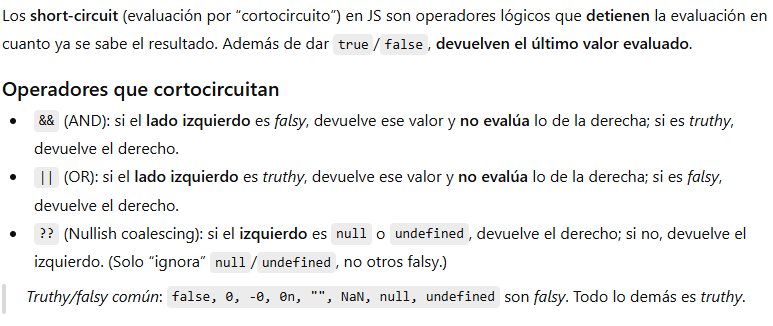
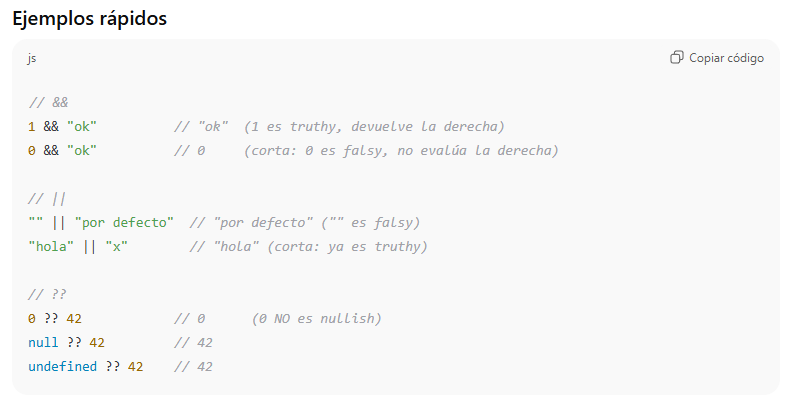
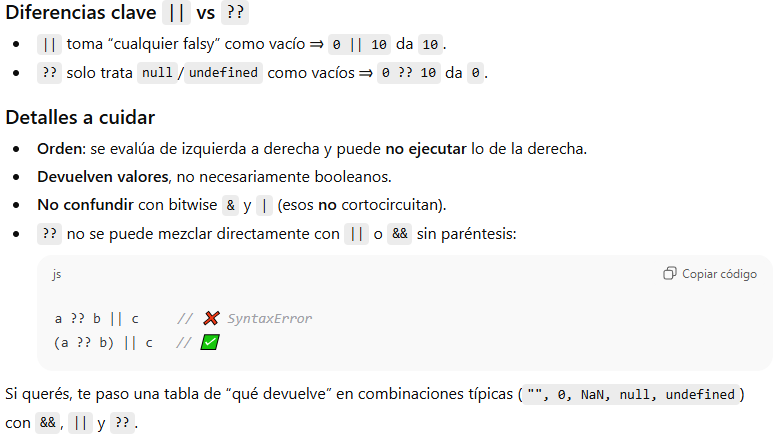


**Operador de comparación que siempre me olvido:**

== 🡪 Compara Valor y NO Tipo (10 == ‘10’ es true)

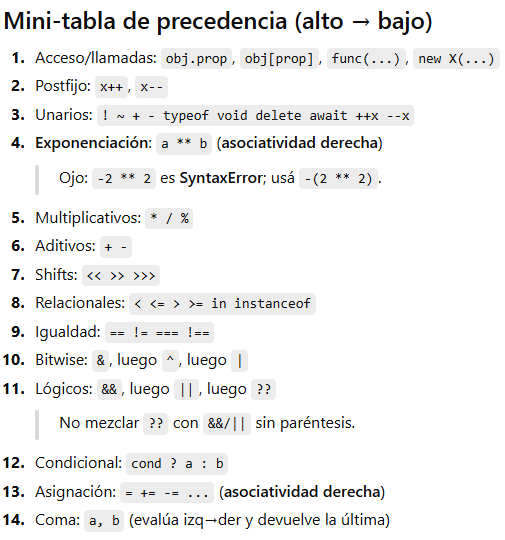
=== 🡪 Compara Valor y NO Tipo (10 == ‘10’ es false)

**Operador Lógico Short circuit:**

**ME SALTIE OPERADORES BITWISE PORQUE NO ENTENDI NADA Y NO LE VI ULTILIDAD**

**Orden de Operaciones**



Los operadores de Control de Flujo en JS son muy parecidos a C#, pero a continuación señalo los que son diferentes:

FOR OF 🡪 foreach en C#



*Nota*: se usa recomendablemente en arrays

FOR IN 🡪 No existe esto en C# 

*Nota*: se usa recomendablemente en objetos

| **Palabra** | **¿Qué hace?** | **¿Dónde se usa?** | **Efecto** |
| --- | --- | --- | --- |
| break | **Sale** inmediatamente del bucle o del switch. | for, while, do…while, switch (JS y C#). | Termina la estructura actual; la ejecución continúa **después** del bucle/switch. |
| continue | **Salta** al **siguiente ciclo** del bucle. | for, while, do…while (JS y C#). | Omite el resto del cuerpo de la iteración; en for ejecuta primero la **actualización** y luego evalúa la condición del siguiente ciclo. |