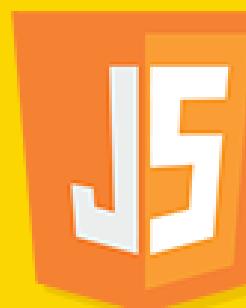
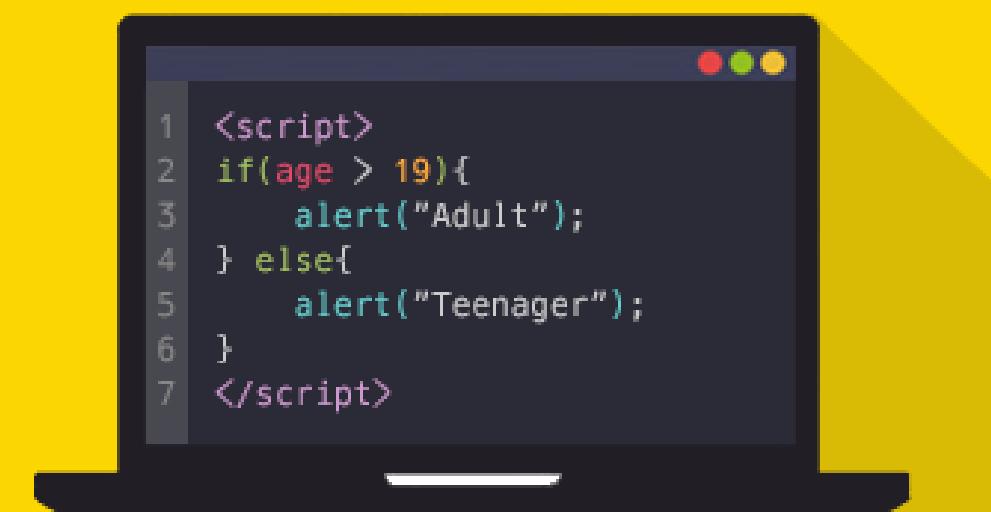


JS

Introducción básica a JavaScript



JavaScript



```
1 <script>
2 if(age > 19){
3     alert("Adult");
4 } else{
5     alert("Teenager");
6 }
7 </script>
```

A dark-themed code editor window showing a single line of JavaScript code. The code contains an if-else conditional statement that checks if the variable 'age' is greater than 19. If true, it alerts 'Adult'. Otherwise, it alerts 'Teenager'. The code is numbered from 1 to 7.

Temas a tratar

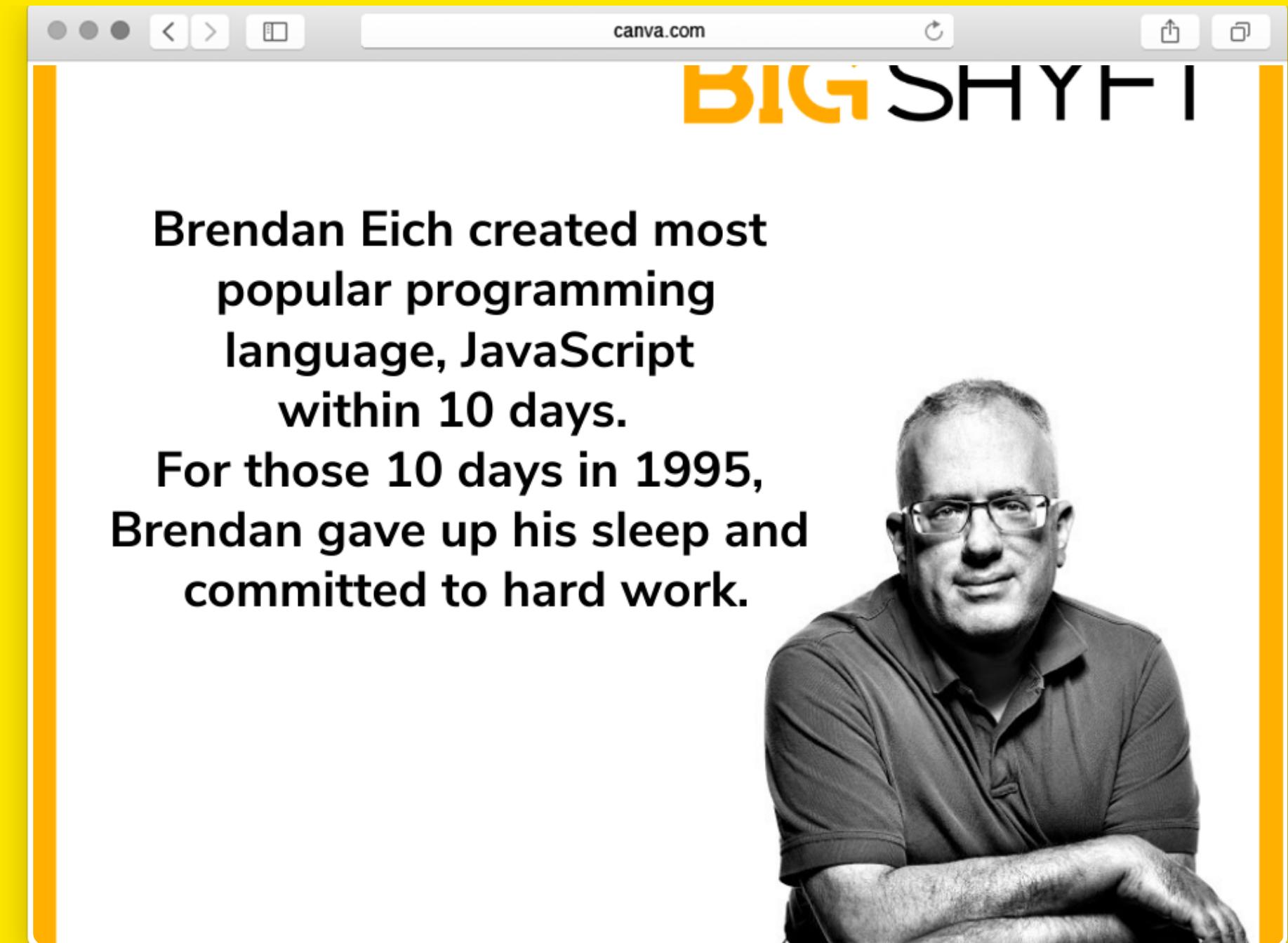
Puntos clave:

- Breve Historia
- Sintaxis Básica
- Operadores
- Estructuras de control
- Funciones Básicas



Breve historia

JavaScript se introdujo en 1995 como una forma de agregar programas a páginas web en el navegador Netscape Navigator. El creador fue Brendan Eich bajo el nombre de "Mocha"



Brendan Eich created most popular programming language, JavaScript within 10 days.

For those 10 days in 1995, Brendan gave up his sleep and committed to hard work.

JS

¿Qué es JS?

Lenguaje de
programación
interpretado



Basado en prototipos
e imperativo



Es débilmente tipado
y dinámico



Es utilizado principalmente en el lado del cliente, implementado como parte de un navegador web permitiendo crear la interacción con el usuario y páginas web dinámicas

Evolución de JavaScript

Historia de JavaScript

Versión de EcmaScript	Año
1	Junio 1997
2	Junio 1998
3	Diciembre 1999
4	Abandonada
5	Diciembre 2009
6	Junio 2015
7	En proceso

Sintaxis Básica

La sintaxis de JS es similar a otros lenguajes de programación como Java y C. Las normas básicas que definen la sintaxis de JS son las siguientes:

- **No se toman en cuenta los espacios en blanco y las nuevas líneas.**
- **Es case-sensitive.**
- **No es necesario definir el tipo de variable.**
- **No es necesario terminar cada sentencia con el carácter de punto y coma.**
- **Se puede incluir comentarios.**

No se toman en cuenta los espacios en blanco y las nuevas líneas.

The diagram shows a snippet of JavaScript code with annotations:

```
1 let language = 'JavaScript' ←  
2 let company = { ←  
3   name: 'EDteam', ←  
4   slogan: 'Nunca te detengas', ←  
5   founded: 2015 ←  
6 }
```

- espacios**: Points to the space before the first character of each line and the space between the assignment operator (=) and the string value.
- tabulacion**: Points to the tab character at the beginning of the third line of the object literal.
- saltos de linea**: Points to the new line character at the end of the first line and the start of the second line.

Es case-sensitive

```
//JS ES Case Sensitive, esto quiere decir que  
//var variable1 es diferente de var Variable1  
var variable1 = 1;  
var Variable = 1;  
// variable1 ≠ Variable
```

Es case-sensitive

```
//JS ES Case Sensitive, esto quiere decir que  
//var variable1 es diferente de var Variable1  
var variable1 = 1;  
var Variable = 1;  
// variable1 ≠ Variable
```

No es necesario terminar cada sentencia con el carácter de punto y coma.

```
//En JS, no es necesario terminar cada sentencia  
//con el caracter de punto y coma  
let nombre = "Carlos"  
let edad = 21;  
let estudiante = true  
  
console.log(nombre, edad, estudiante);
```

Se puede incluir comentarios

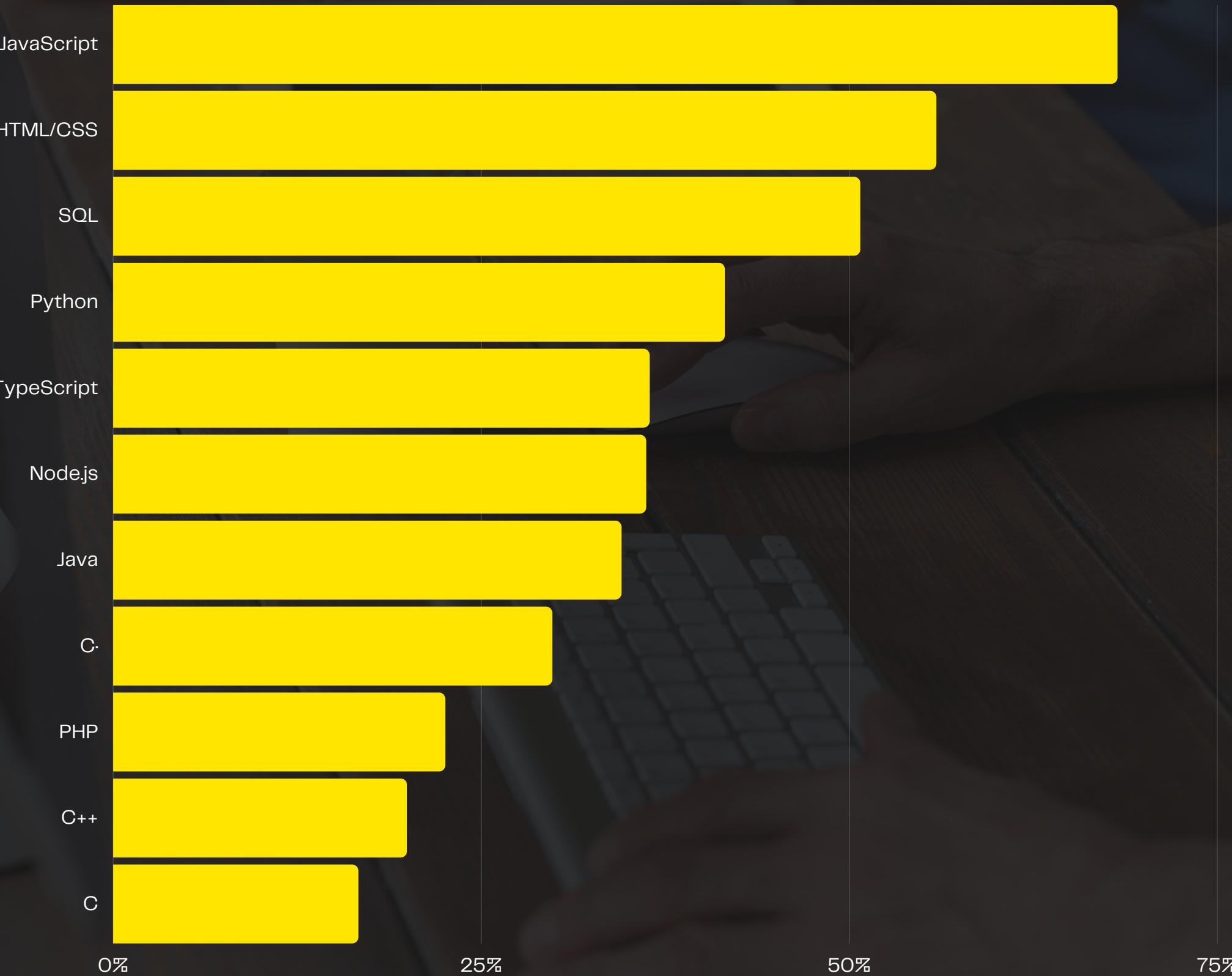
```
//En JS, Se pueden incluir comentarios
let nombre = "Carlos"
// Comentario de una linea
let edad = 21;
/*
    Comentario
    de
    n
    lineas
*/
let estudiante = true
```

Lenguajes de programación más populares 2022

JS se mantiene como el lenguaje de programación más popular del mundo

TALLER JAVASCRIPT BÁSICO 2022

JS



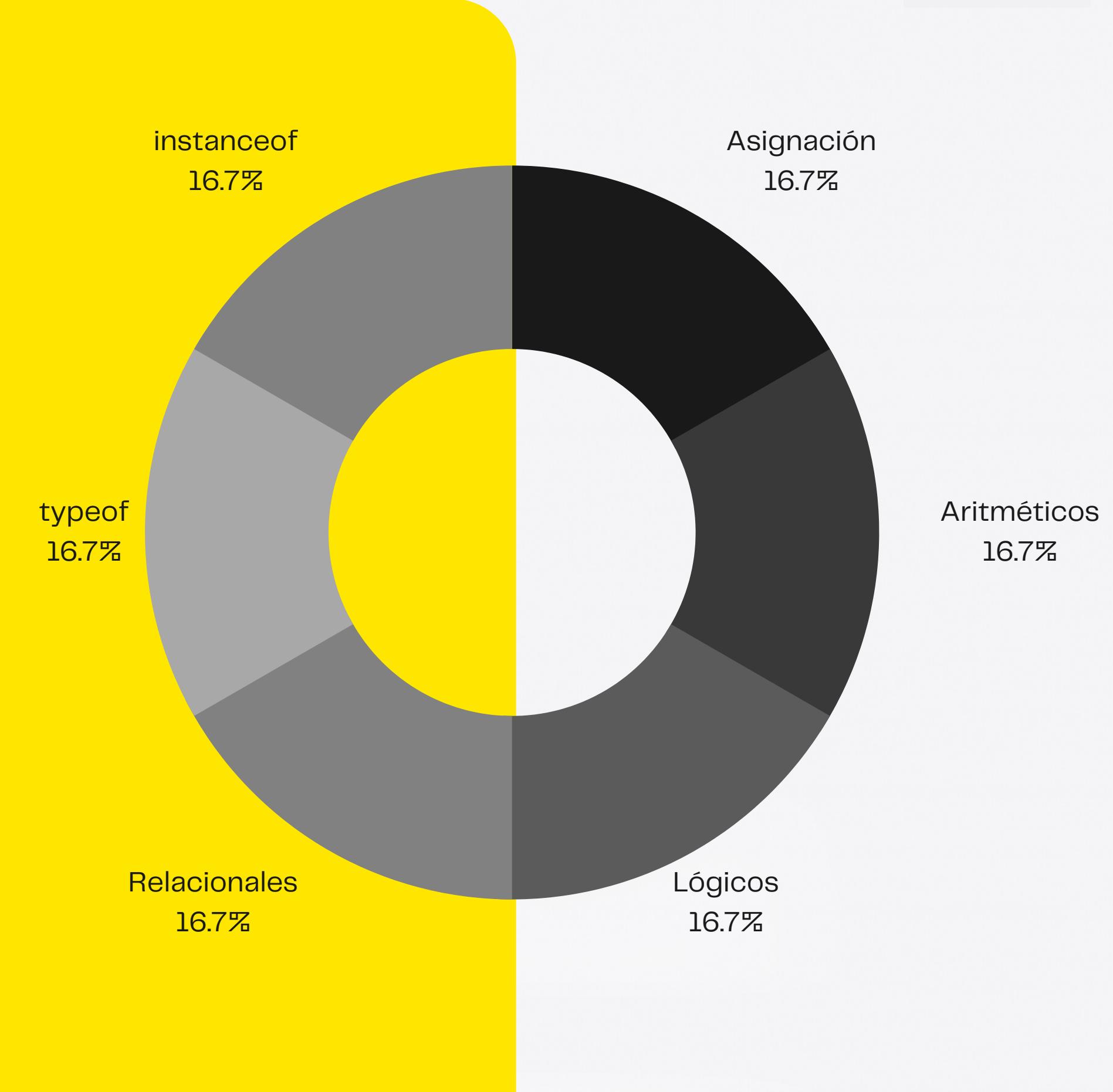


JS

Operadores

Permiten manipular el valor de las variables, realizar operaciones matemáticas con sus valores y comparar diferentes variables.

De esta forma, los operadores permiten a los programas realizar cálculos complejos y tomar decisiones lógicas en función de comparaciones y otros tipos de condiciones.



Tipos de operadores

```
var numero1 = 3;
```

Asignación

Se utiliza para guardar un valor en específico

```
ultado = numero1 / numero2;
ultado = 3 + numero1;
ultado = numero2 - 4;
ultado = numero1 * numero2;
ultado = numero1 % numero2;
```

Aritméticos

Se utiliza para realizar manipulaciones matemáticas

expr1 && expr2

expr1 || expr2

!expr

Lógicos

Se utilizan para tomar decisiones sobre las instrucciones a ejecutar

Tipos de operadores

```
sultado = numero1 >= num  
sultado = numero1 <= num  
sultado = numero1 == num  
sultado = numero1 != num
```

Relacionales

Se utilizan para relacionar variables de cualquier operación compleja

```
typeof myFunction
```

Typeof

Se emplea para determinar el tipo de dato que almacena una variable

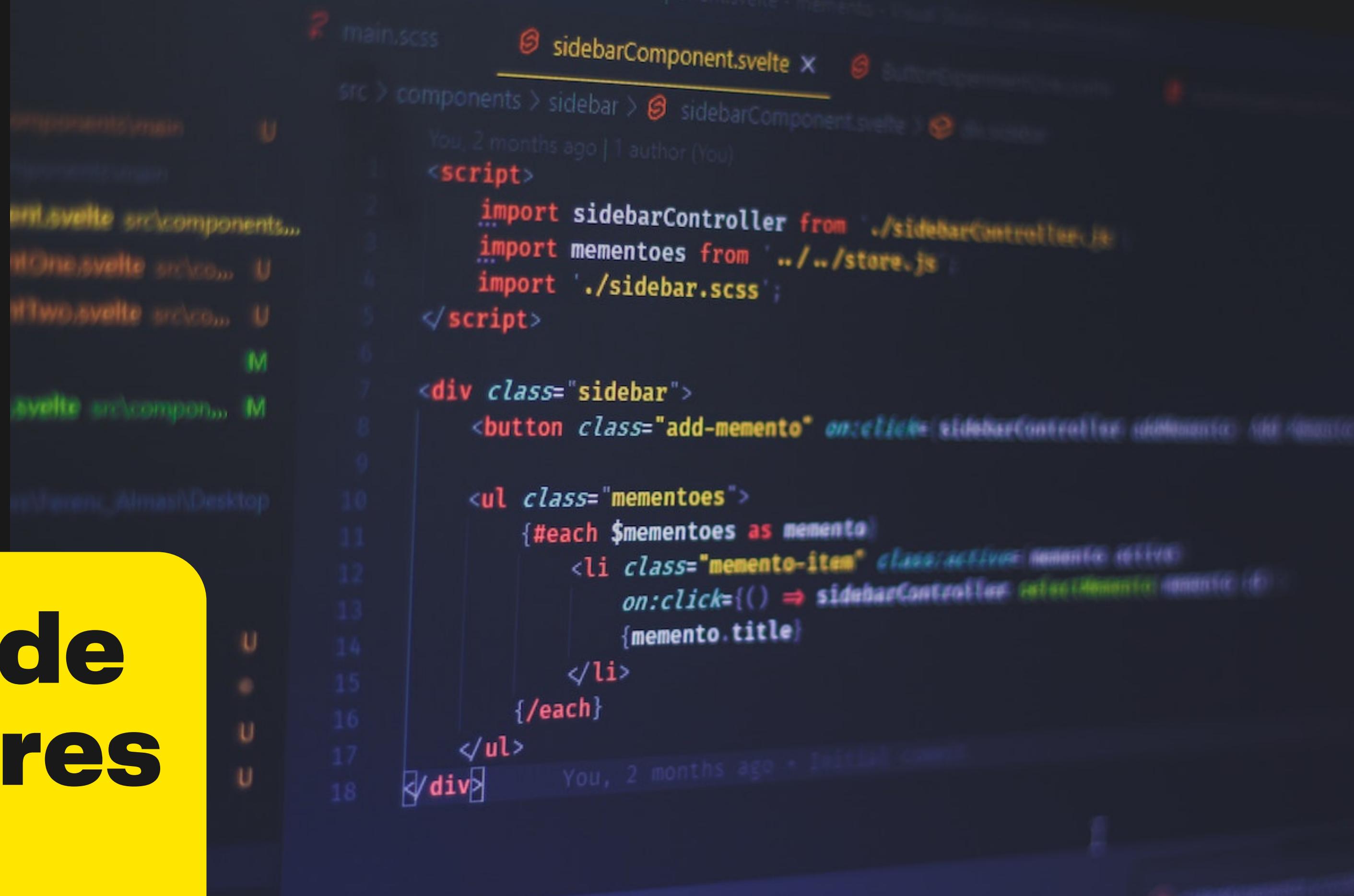
```
riable1 instanceof String
```

Instanceof

Se emplea para determinar la clase concreta de un objeto

JS

Ejemplo de Operadores



The screenshot shows a code editor with a dark theme. On the left, there's a sidebar with file navigation. The main area displays Svelte component code. The code includes imports for `sidebarController`, `mementoes`, and a local `sidebar.css`. It features a `<script>` block and a `<div>` block containing a `<button>` and a `` for displaying mementoes. The `` uses a `#each` loop to iterate over `$mementoes`, rendering `` elements with `on:click` events.

```
main.scss
sidebarComponent.svelte
src > components > sidebar > sidebarComponent.svelte
You, 2 months ago | 1 author (You)

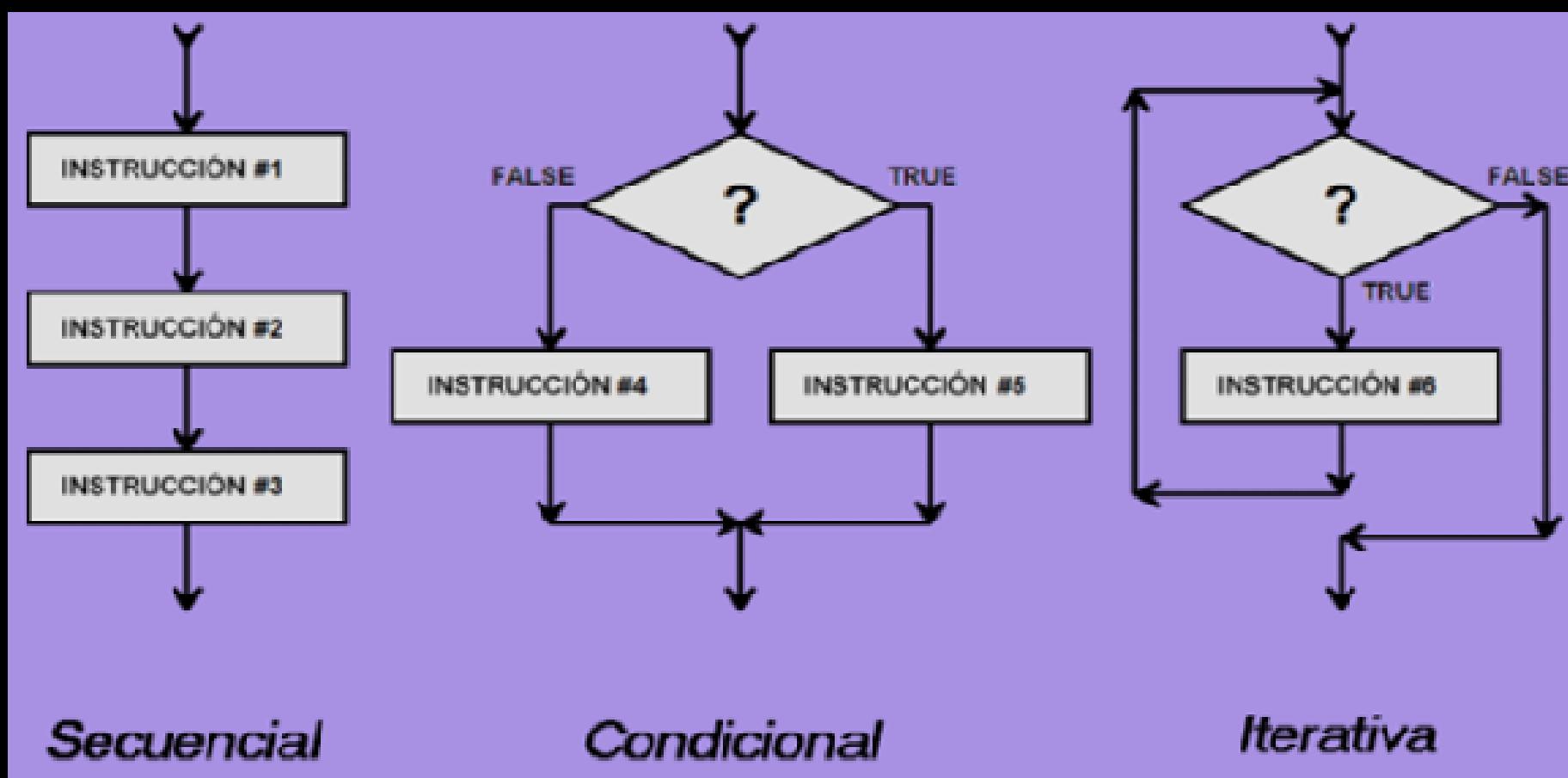
<script>
    import sidebarController from './sidebarController.js';
    import mementoes from '../store.js';
    import './sidebar.css';

</script>

<div class="sidebar">
    <button class="add-memento" on:click=sidebarController.addMemento></button>

    <ul class="mementoes">
        {#each $mementoes as memento}
            <li class="memento-item" class:active={memento.active}>
                on:click={() => sidebarController.setValue(memento.title)}
                {memento.title}
            </li>
        {/each}
    </ul>
</div>
```

Estructuras de control en JS

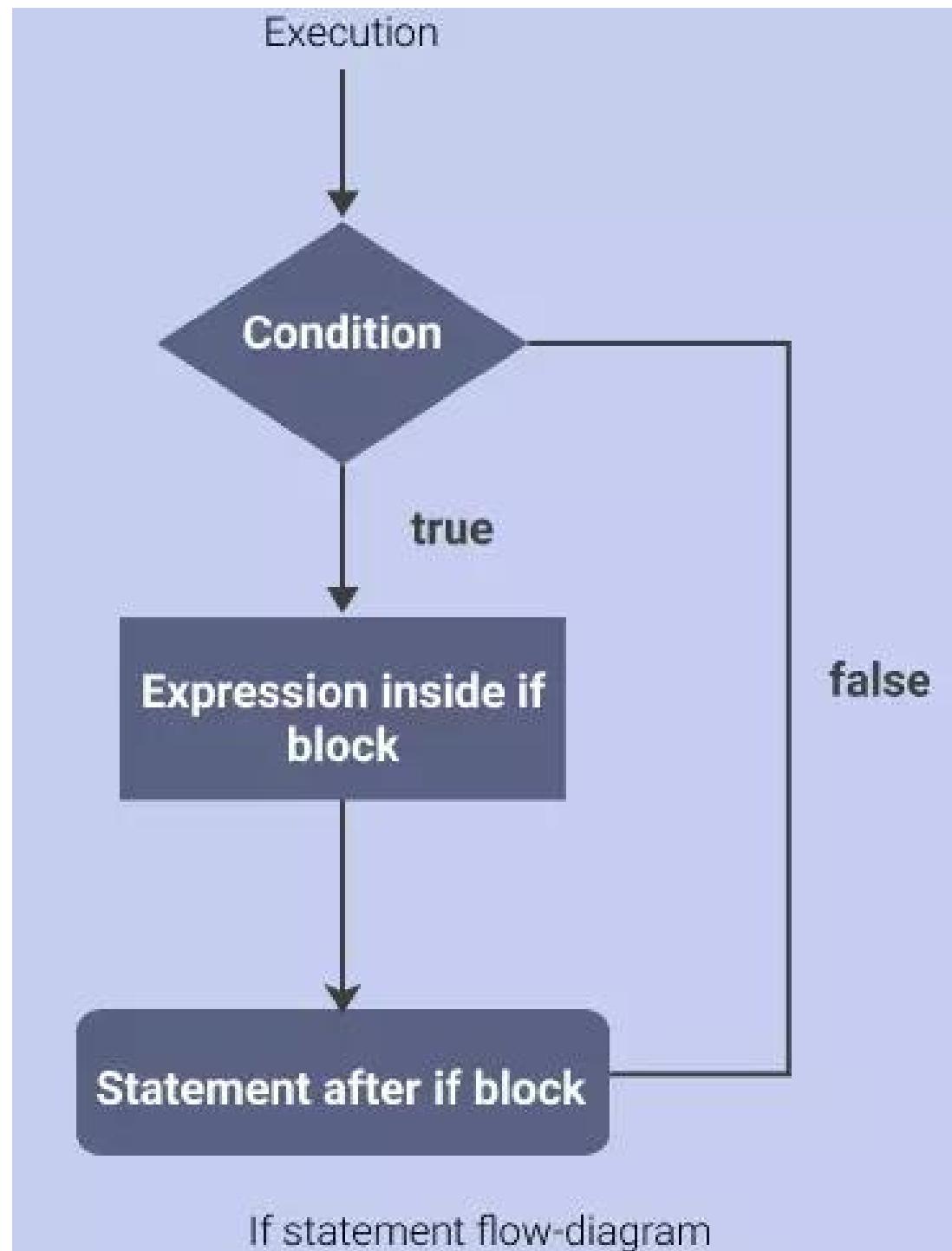


Son instrucciones necesarias para realizar tareas distintas dependiendo del estado de nuestras variables o realizar un mismo proceso muchas veces sin escribir las mismas líneas de código una y otra vez.

Estructura de control

IF....ELSE

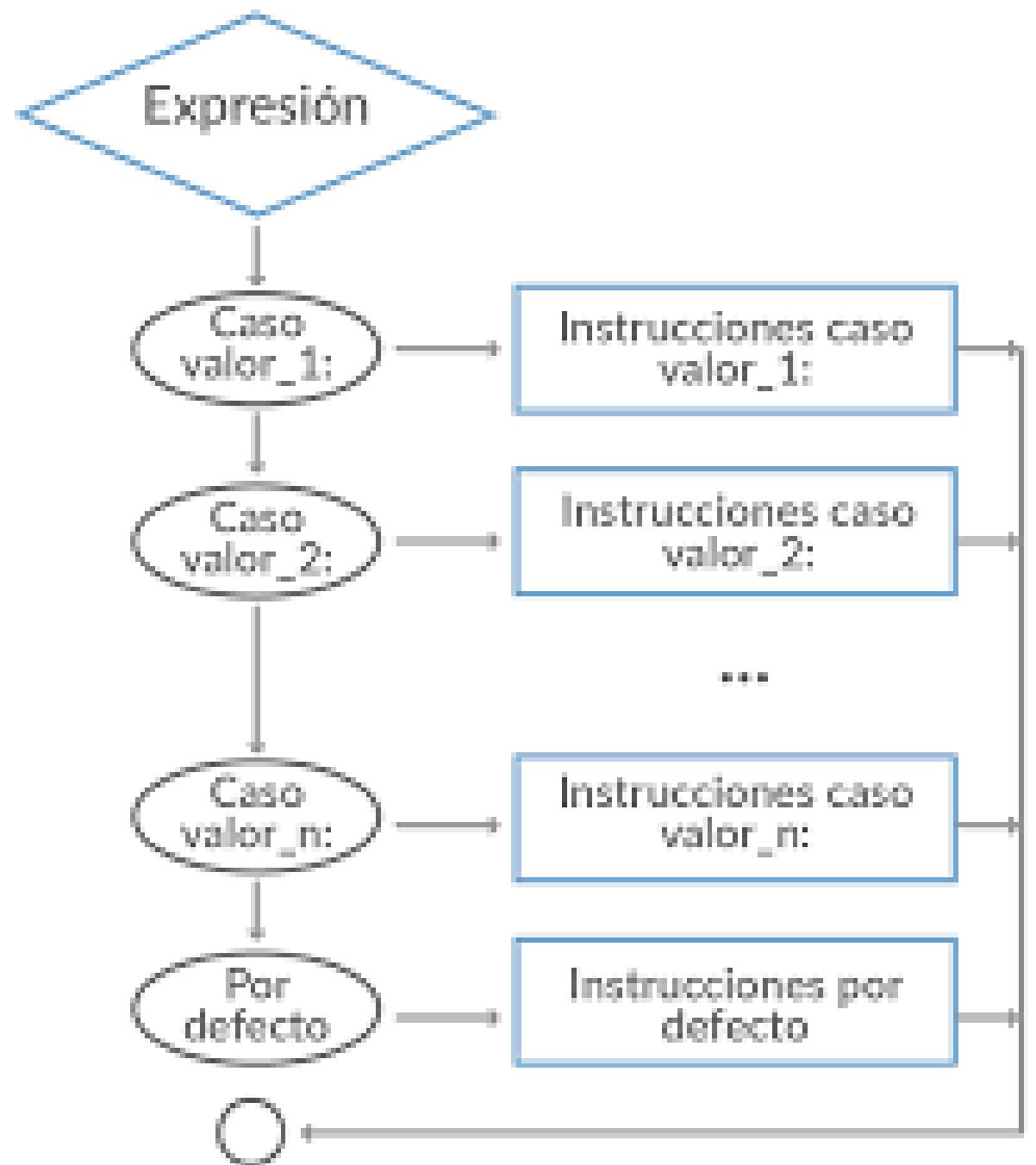
JS



```
let condicion = true;  
  
if(condicion) {  
    console.log("Condicion verdadera");  
}else {  
    console.log("condicion falsa")  
}
```

Estructura de control SWITCH....

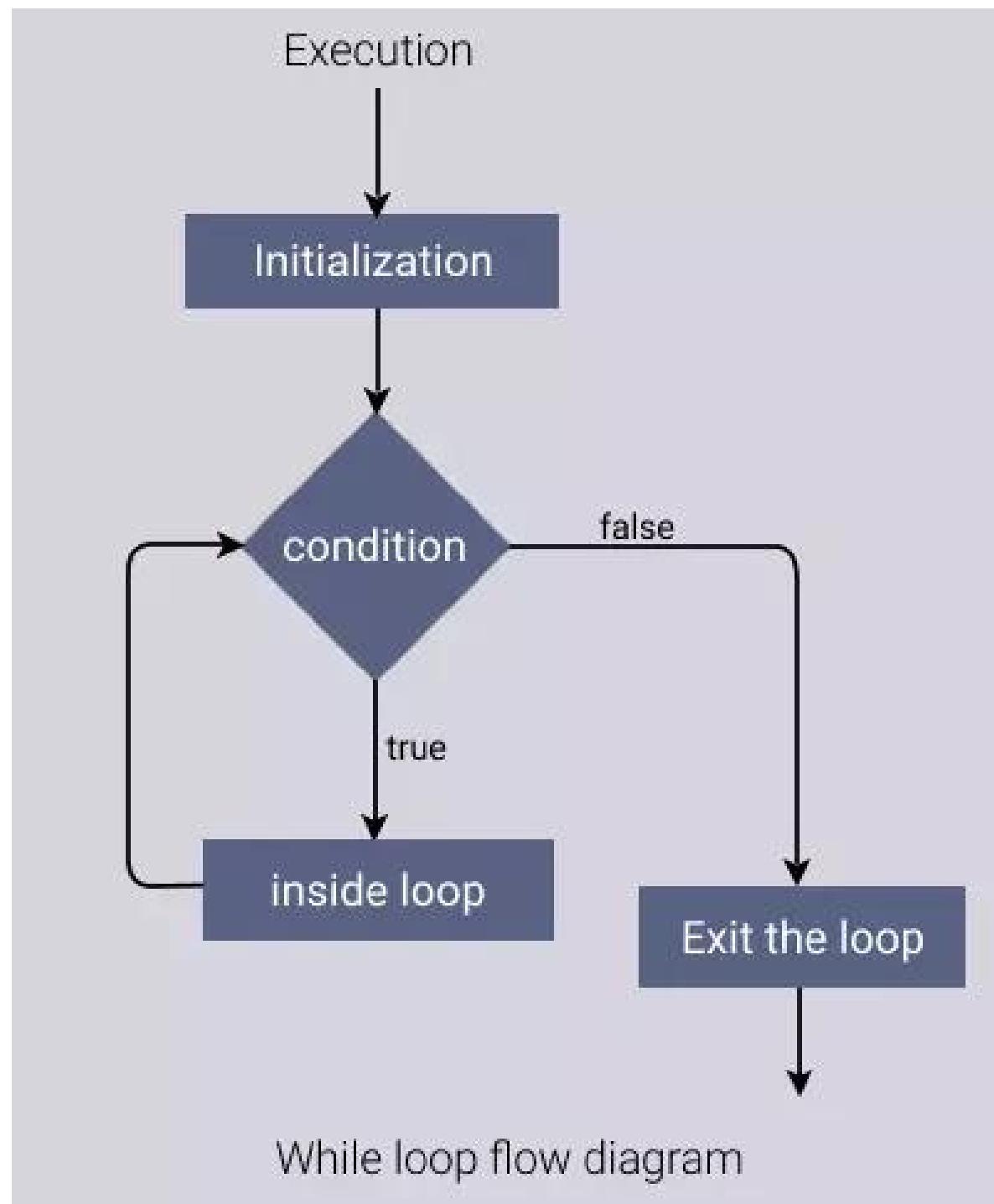
JS



```
let condicion = 2;  
  
switch (numero) {  
    case 1:  
        numeroTexto = "Numero uno";  
        break;  
    case 2:  
        numeroTexto = "Numero dos";  
        break;  
    case 3:  
        numeroTexto = "Numero tres";  
        break;  
    default:  
        numeroTexto = "Numero no encontrado";  
        break;  
}  
console.log(numeroTexto);
```

Estructura de control WHILE....

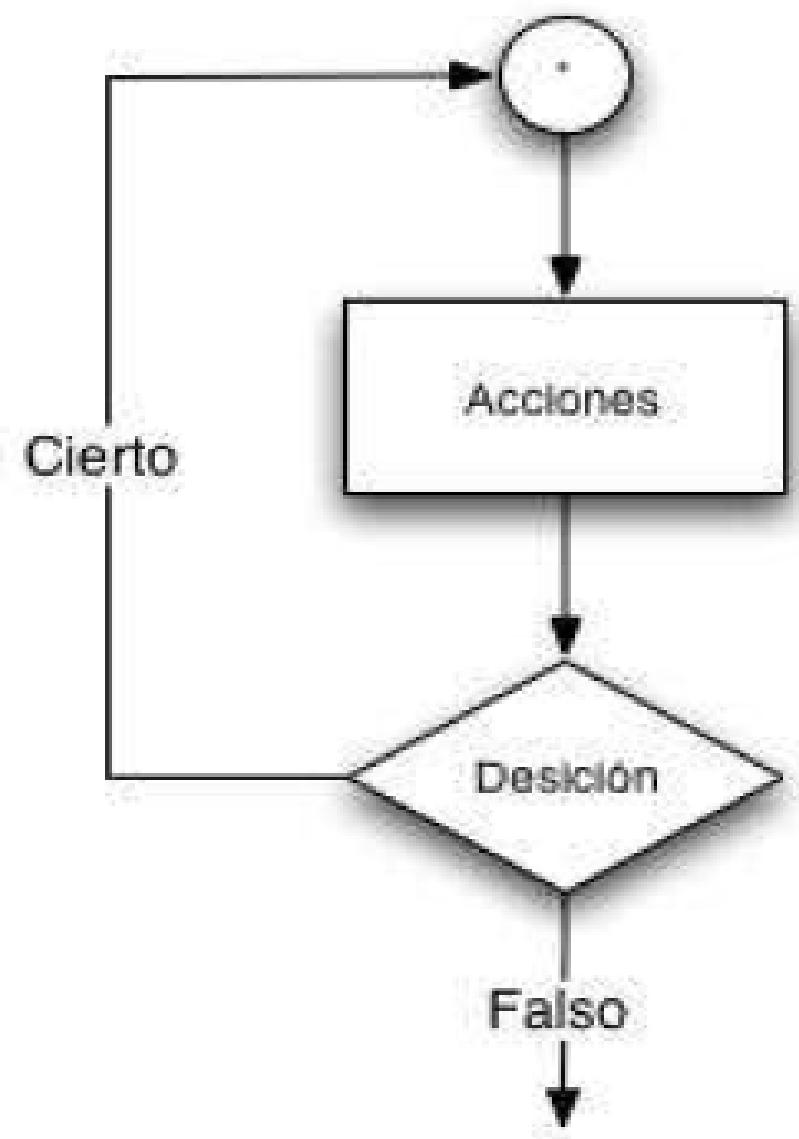
JS



```
let contador = 0;
while (contador < 10) {
  console.log(++contador); // 1,2,3,4,5,6,7,8,9,10
}
console.log("Fin ciclo while");
```

Estructura de control DO...WHILE....

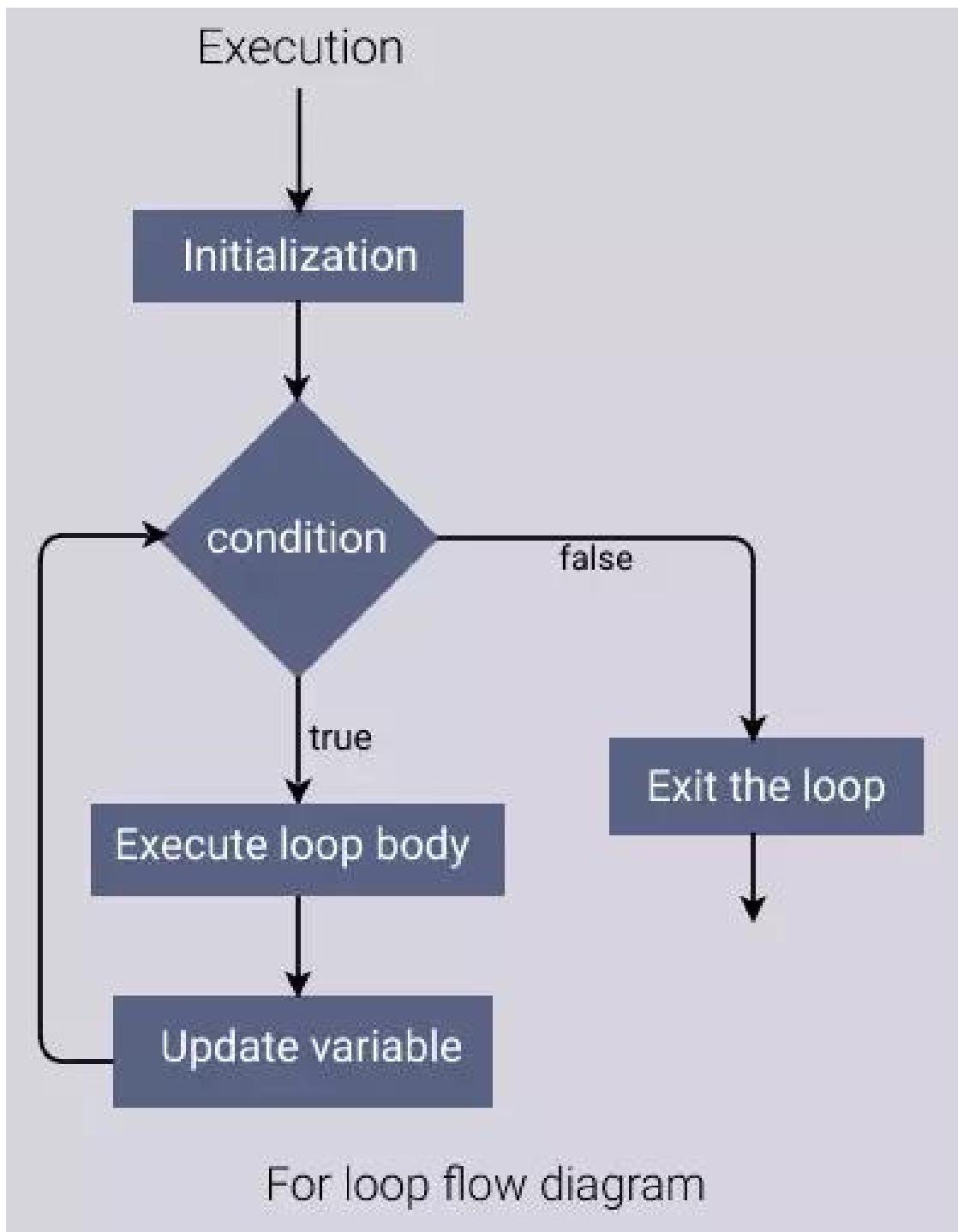
JS



```
let contador = 0;  
do {  
    console.log(contador++); //0,1,2,3,4,5,6,7,8,9.  
} while (contador < 10);
```

Estructura de control FOR....

JS



```
for (let contador = 0; contador < 10; contador++) {  
  if (contador % 2 === 0) {  
    console.log(contador);  
    break;  
  }  
}  
console.log("Fin ciclo for");
```

For loop flow diagram

Funciones en JS

¿Qué es una función?

Es una secuencia con nombre de declaraciones que realiza una operación específica.

¿Por qué usar un función?

Sencillamente para **dividir un programa** en piezas más pequeñas y reutilizables.

Construcción básica de una función

```
function nombreFuncion(param1, param2, ...) {  
    // declaraciones  
    return;  
}
```

return

Las funciones no solo pueden recibir valores o datos, también pueden o no devolver los valores.

`()` \Rightarrow `{ }`

Función de flecha en JavaScript

Es una de las características introducidas en ES6 para una mejor escritura en JS.

Función Flecha

`() => {}`

Es una alternativa compacta a una función tradicional



No tiene sus propios enlaces a this o super y no se debe usar como método.



No tiene argumentos o palabras clave.



No apta para los métodos call, apply y bind, que normalmente se basan en establecer un ámbito o alcance.



No se puede utilizar como constructor.



No se puede utilizar yield dentro de su cuerpo.



JS

Gracias!!!!

**Construye algo que
el encante a 100
personas, no solo
algo que le guste a
1000 personas.**

Brian Chesky