Estructuras de control

Una estructura de control es una construcción de código que dirige la ejecución de nuestro código. Determina la ruta que seguirá el proceso en ciertas condiciones.

If...else...else if

Forma general / Pseudocódigo:

```
if ( BOOLEAN ) {

   // code

} else if ( BOOLEAN ) {

   // code

} else {

   // code

}
```

Ejemplos:

```
var result = ''; if (a > 2) {
  result = 'a es mayor que 2';
}
if (a > 2) {
  result = 'a es mayor que 2';
} else {
  result = 'a NO es mayor que 2';
}
if (a > 2 || a < -2) {
  result = 'a no está entre -2 y 2';
} else if (a === 0 && b === 0) {
  result = 'ambos a y b son ceros';
} else if (a === b) {
  result = 'a y b son iguales';
} else {
  result = 'nadie cumple';
}
```

- La sentencia `if`, `else if` o `else`
- Una condición entre paréntesis.
- Esta condición siempre devolverá un booleano y puede contener:
  - Una operación lógica: `!`, `&&` o `||`.
  - Una comparación como `===`, `!==`, `>` y otros.
  - Cualquier valor o variable que pueda ser convertido a Booleano.
  - Una combinación de todas.
- El bloque de código a ejecutar si se cumple la condición.

 https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Sentencias/if...else

Switch

Forma general / Pseudocódigo:

```
switch ( VALUE ) {
   when VALUE_1:
     CODE;
     break;
   when VALUE_2:
     CODE;
     break;
   default: CODE;
}
```

Ejemplos:

```
var ageName = 'joven adulto', ageRange;
switch ( ageName ) {
   case 'infancia':
     ageRange = [3, 12];
     break;
   case 'adolescente':
     ageRange = [13, 18];
     break;
   case 'joven adulto':
     ageRange = [18, 30];
     break;
   case 'adulto':
```

```
    ageRange = [30, Infinity];
}
ageRange; // [ 18, 30 ]
```

- La sentencia switch
- Una expresión entre paréntesis.
- Esta expresión normalmente será una variable, pero puede ser cualquier expresión que devuelva un valor.
- Cierto número de bloques `case` entre corchetes.
- Cada sentencia case va seguida de una expresión.
- El resultado de esta expresión se compara con la expresión que hay después del switch. Si la comparación de igualdad devuelve `true`, se ejecuta el bloque que hay tras este case
- Puede (y debe) haber una sentencia `break` al final de cada bloque case. Estos break provocan la salida del `switch` (de esta manera nos aseguramos de ejecutar un solo bloque `case`)
- También puede (y debe) haber una sentencia `default` que es seguida de un bloque de código que se ejecuta si ninguno de los case es evaluado a `true`

https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Sentencias/switch

Operación ternaria

Es un operador porque devuelve un valor. Se puede usar **en cualquier lugar** donde se espera un valor

Forma general / Pseudocódigo:

```
BOOLEAN ? VALUE_WHEN_TRUE : VALUE_WHEN_FALSE
```

Ejemplos:

```
var a = ( n > 5 ) ? 'n es mayor que 5' : 'n es menor igual que 5';
function log( msg ) {
   console.log( msg );
}
var n = 3; log( ( n > 5 ) ? 'n es mayor que 5' : 'n es menor igual
que 5' ); //> n es menor igual que 5
```

while

Forma:

```
while ( BOOLEAN ) {
   // code
}
```

Ejemplo:

```
var a = 0; while ( a < 5 ) {
   console.log( a );
   //> 0, 1, 2, 3, 4   a++;

}
```

- La sentencia `while` va seguida de una condición entre paréntesis y un bloque de código entre corchetes.
- Mientras la condición se evalúe a true, el código se ejecutará una y otra vez.
- El número de repeticiones dependerá del resultado de evaluar una condición, antes (o después) de cada iteración

https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Sentencias/while

do-while

Forma general / Pseudocódigo:

```
do {
   // code
} while ( BOOLEAN );
```

Ejemplo:

```
var a = 0; do {
   console.log( a );
   //> 0, 1, 2, 3, 4
   a++;
} while ( a < 5 );
```

El bucle do-while es una pequeña variación del bucle while. La sentencia `do` va seguida de un bloque de código y una condición después del bloque. Esto implica que el bloque de código se va a ejecutar siempre, al menos una vez, antes de evaluar la condición.

https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Sentencias/do...while

for

Forma general / Pseudocódigo:

```
for ( CODE; BOOLEAN; CODE ) {
   // code
}
```

Ejemplo:

```
var a; for ( a = 0; a < 5; a++ ) {
   console.log( a );
   //> 0, 1, 2, 3, 4

}
```

La estructura del bucle `for` tiene 3 partes claramente diferenciadas (separadas por `;`)

- Inicialización (`var i = 0`): Código que es ejecutado *antes* de entrar en el bucle *[O]*
- Evaluación (`i < 100`): Mientras evalúe a `true` seguimos con el bucle *[C]*
- Incremento (`i++`): Código que es ejecutado después de cada iteración *[++]*

https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Sentencias/for

for-in

Forma general / Pseudocódigo:

```
for ( VARIABLE in OBJECT ) {
   // code
}
```

Ejemplos:

```
var a, obj = { a: 1, b: 2, c: 3 };
for ( a in obj ) {
   console.log( a );
   //> 'a', 'b', 'c'
}
var a, arr = [ 'a', 'b', 'c', 'd' ];
for ( a in arr ) {
   console.log( a );
   //> 0, 1, 2, 3
```

```
}
```

El bucle for-in es utilizado para recorrer los elementos de un objeto (normalmente de un objeto Array)

https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Sentencias/for...in