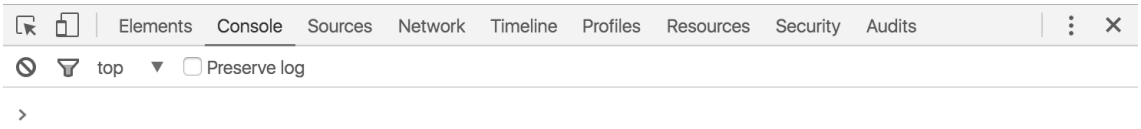
ESTRUCTURAS DE CONTROL EN JAVASCRIPT

"No traten de guiar al que pretende elegir por sí su propio camino".

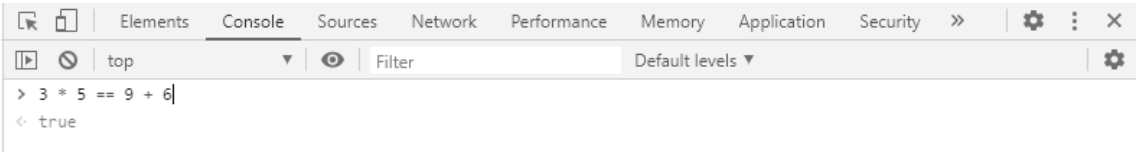
(William Shakespeare)

¿CÓMO ABRIR LA CONSOLA DEL DESARROLLADOR?

Los primeros grandes misterios que tenemos como desarrolladores es como probar cierto código o como descubrir un error. Por suerte los navegadores vienen con herramientas integradas para trabajar con JavaScript y otras tecnologías web que nos facilitan la vida. Por ejemplo, para acceder a la consola de Google Chrome y probar nuestro código en JavaScript es necesario presionar las teclas CTRL+SHIFT+I, lo mismo si quiero acceder desde Mozilla Firefox o desde Microsoft Edge.



Estas herramientas incluyen una consola similar a un Shell de un Sistema Operativo. Por ejemplo, podemos escribir el siguiente código 3 * 5 == 9 + 6 detrás del símbolo > y el resultado será true. Es decir, arrojamos una expresión aritmética para comparar dos expresiones (la izquierda con la derecha) usando el operador de igualdad, luego de la evaluación tenemos el valor booleano true (es decir verdadero).



Por supuesto, las expresiones en JavaScript pueden ser mucho más sofisticadas que esto. Por ejemplo podemos usar el método floor() del objeto Math que elimina la parte fraccionaria de un numero con decimales redondeando hacia abajo. Podemos hacer la siguiente prueba:

```
> Math.floor(17 / 2)
```

Este devolverá el valor 8, ya que si 17 dividido 2 es 8.5 y al redondearlo y llevarlo hacia abajo devuelve 8.
Otro elemento importante para trabajar con la consola es mandar avisos directamente desde nuestra aplicación. Para ello podemos usar el siguiente comando:

```
console.log(MENSAJE);
```

Esto nos va a permitir identificar de forma rápida lo que está ocurriendo en la página.:

```
let someNumber=30;
let sum=someNumber+5;
console.log(sum);
```

Este pequeño script muestra por consola el valor de la variable sum. Si nuestra pantalla hay un exceso de mensajes, podemos limpiar la consola de nuestro navegador simplemente haciendo clic con el botón derecho y seleccionando clear console o también lo podemos hacer a través de código tecleando clear().

ESTRUCTURAS DE CONTROL

Hasta aquí hemos obtenidos programas que son una simple sucesión lineal de instrucciones. Pero que sucede si nuestro programa debe tomar decisiones en base a una condición o si necesitamos repetir una cierta cantidad de veces un conjunto de instrucciones. Para realizar este tipo de programas son necesarias las estructuras de control de flujo, que son instrucciones del tipo "si se cumple esta condición, hazlo; si no se cumple, haz esto otro". También existen instrucciones del tipo "repite esto mientras se cumpla esta condición".

ESTRUCTURA IF

La estructura más utilizada en JavaScript y en la mayoría de lenguajes de programación es la estructura if. Se emplea para tomar decisiones en función de una condición. Si la condición se cumple (es decir, si su valor es true) se ejecutan todas las instrucciones que se encuentran dentro de un bloque. Si la condición no se cumple (es decir, si su valor es false) no se ejecuta ninguna instrucción contenida en el bloque anterior, pero continúan ejecutando las instrucciones fuera del if. Para poder trabajar con las condiciones debemos primero conocer los operadores.

OPERADORES DE COMPARACION

Un operador es un símbolo en una expresión que representa una operación aplicada a los valores sobre los que actúa. Los valores sobre los que actúa un operador se llaman operandos. Un operador binario es el que tiene dos operandos, mientras que un operador unario es el que tiene sólo uno.
Los operadores de comparación devuelven un valor de tipo boolean (verdadero o falso) que indica el resultado de la comparación. Algunos operadores de comparación son:

- El **igual a** == (no confundir con el = de las asignaciones)
- El **distinto a** !=
- El **mayor que** >
- El **mayor o igual que** >=
- El **menor que** <
- El **menor o igual que** <=
- La **igualdad estricta** ===
- La **desigualdad estricta** !==

IGUALDAD E IGUALDAD ESTRICTA

La diferencia entre == (igualdad) y === (igualdad estricta) es que el operador de igualdad convertirá los valores de distintos tipos antes de comprobar la igualdad. Por ejemplo, el resultado de comparar la cadena "1" con el número 1 es true. El operador de igualdad estricta, por otra parte, no convertirá los valores a tipos diferentes, por lo que la cadena "1" y el número 1 no se comparan como iguales.

OPERADORES ARITMÉTICOS

Las operaciones aritméticas son las que operan sobre valores numéricos y entregan otro valor numérico como resultado. Algunos operadores aritméticos son:

- La **suma +**
- La **resta** -
- La **multiplicación** *
- La **división** /
- El **módul**o % (resto de la división);
- **Incremento en uno ++**
- **Decremento en uno** --

OPERADORES LÓGICOS O BOOLEANOS

Los operadores booleanos o lógicos se utilizan para formar expresiones que sólo pueden tener, como resultado, un valor booleano. Hay tres operadores lógicos:

- El **operador !** es un operador unario, es decir, sólo se aplica a un operando. Su resultado es cambiar el valor del operando al que se aplica, es decir, si es verdadero, lo cambia a falso y viceversa.
- El **operador OR o ||** se puede aplicar a dos o más operandos. El resultado es verdadero si uno de los operandos o expresiones son verdaderas.
- El **operador AND o &&** igual que el anterior puede operar con varios o más operandos. En este caso todos los operandos tienen que ser verdadero para que dé un resultado true.

ESTRUCTURA IF-ELSE

En ocasiones, las decisiones que se deben realizar son del tipo "si se cumple esta condición, hazlo; si no se cumple, haz esto otro". Para este caso se debe crear otro bloque de instrucciones en el caso de la condición sea falsa. Este bloque ira después de la palabra reservada else.

```
if(condición){
        instrucciones a ejecutar si se cumple la condición
}
else
{
        instrucciones a ejecutar si NO se cumple la condición
}
```

Ahora desarrollaremos un pequeño ejemplo donde le pasamos el talle de una remera (S, M, L o XL) y nos devuelve el color de la remera que tiene la tienda virtual:

```
function colorRemera(talle){
        // Solo talles: S, M, L Y XL.
        let color;
        if (talle=="S")
                color="Blanca";
        else if (talle=="M" || talle=="L")
               color="Negra";
        else
                color="Roja";
        return color;
}
```

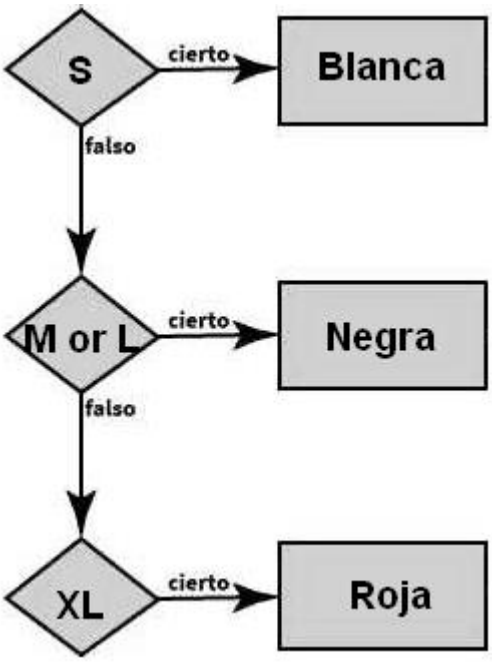


Figura 10.1. Diagrama de Flujo del ejemplo anterior (Estructura if).

ESTRUCTURA FOR

Cuando se sabe a priori el número de veces que se tiene que repetir una operación o tarea se utiliza la estructura for Esta estructura crea un bucle que ejecuta una o un conjunto de instrucciones mientras cierta condición se evalúe como verdadera. Cuando la condición se evalúa como false, la ejecución continúa con la sentencia posterior al bucle for.
Su estructura de control está compuesta de tres declaraciones:

```
for ((expresión inicialización); (condición de salida); (incremento o
decremento))
{
   instrucciones a ejecutar si se cumple la condición
}
```

La primera expresión de inicialización le asigna un valor inicial a una variable. La siguiente es la expresión de condición que indica cuando terminara el bucle. La tercera parte representa un incremento o decremento que debe adoptar la variable declarada en la primera expresión

Ahora desarrollaremos un ejemplo donde le pasamos la cantidad de veces que queremos repetir una frase.

```
function copiarFrase (cantidad){

let frase="Aquí me pongo a cantar <br>
Al compás de la vigüela,<br>
Que el hombre que lo desvela<br>
Una pena extraordinaria<br>
Como la ave solitaria<br>
Con el cantar se consuela";

for (let i = 0; i < cantidad; i++) {
    document.write(frase);
    document.write('<br>');
    document.write('--');
    document.write('<br>');
}
}
```

La declaración i es una expresión de inicialización, es decir, se asigna un valor inicial a una variable, que generalmente se la llama i (índice o iterador). La condición de salida es que la cantidad de veces a repetir sea menor a un valor cantidad (cantidad es un argumento de la función que contiene la cantidad de veces que se quiere repetir la frase) en el caso de que el valor i sea superado se sale del bucle for por que la condición se transforma en falsa.Como la variable i se ha inicializado a un valor de 0 y la condición para salir del bucle es que i sea menor que la cantidad que se le pase a la función, si no se modifica el valor de i de alguna forma, el bucle se repetiría indefinidamente. Por ese motivo, es imprescindible indicar un incremento de i para poder controlar el bucle.
El código HTML que invoca la función es el siguiente:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <script src="js/app.js"></script>
</head>

<body>
        <button>Repetir Frase</button>
        <script>
            let btn = document.querySelector('button');
            btn.addEventListener('click', function(){copiarFrase(4)});
        </script>
</body>
</html>
```

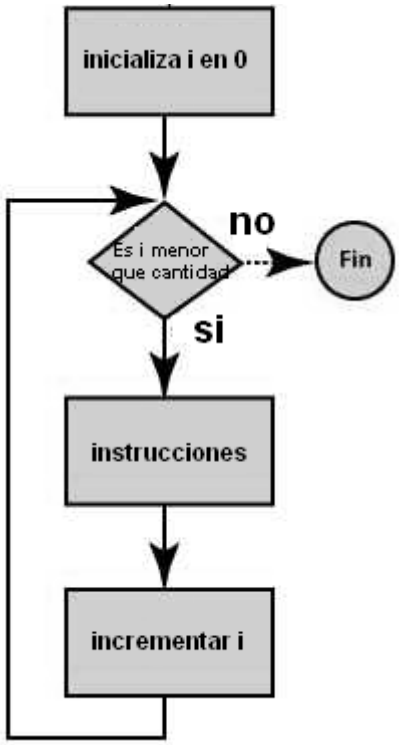


Figura 10.2. Diagrama de Flujo del ejemplo anterior (estructura for).

ESTRUCTURA WHILE

La estructura de control while es otra estructura de repetición mientras se cumple una condición. La sintaxis en JavaScript seria la siguiente:

```
while (condicion) {
   instrucciones a ejecutar
}
```

Para usar esta estructura (a diferencia del for visto anteriormente) hay que tener en cuenta que, en este tipo de estructura iterativa, la variable se tiene que inicializar fuera del bucle y su incremento, dentro del bucle while. O sea que ahora tendríamos la siguiente sintaxis:

```
(expresión inicialización)
while (condicion) {
   instrucciones a ejecutar
         .
         .
         .
   (incremento o decremento)
}
```

Tanto la inicialización como la modificación de la variable de inicialización son muy importantes, cualquier error podría llevar a comportamientos extraños de la aplicación. La aplicación anterior usando la estructura while quedaría de la siguiente forma:

```
function copiarFrase (cantidad){

let frase="Aquí me pongo a cantar <br> Al compás de la vigüela,<br>
Que el hombre que lo desvela<br>
Una pena extraordinaria<br>
Como la ave solitaria<br>
Con el cantar se consuela";

let i = 0;
while (i<cantidad){
    document.write(frase);
    document.write('<br>');
    document.write('--');
    document.write('<br>');
    i++;
}
}
```

Como se puede ver la variable se inicializa antes de usar el while. La condición de salida se encuentra justo en el while y dentro del bloque me encargo de incrementar la variable de inicialización.