TALLER EVALUATIVO LÓGICA CON JAVASCRIPT

DANIEL ACEVEDO GONZALEZ
MAURICIO MORALES VIDAL

ANALISIS Y DESARROLLO DE SOFTWARE

N° DE FICHA 2952663

GERMAN ALBERTO ANGARITA HENAO

27/06/2024

Tabla de contenido

Tipo de datos:

JavaScript maneja varios tipos de datos, los cuales son:

Primitivos:

Se conocen como primitivos porque sus valores suelen contener una sola cosa:

- Number: Representa tanto números enteros como de punto flotante.
- BigInt: Representa números enteros de longitud arbitraria, es creado al agregar n al final de un entero.
- String: Representa una cadena de caracteres y debe colocarse entre comillas.
- Boolean: Representa solo dos valores posibles, true y false.
- Null: Representa un valor especial como “nada”, “vacío” o “valor desconocido”.
- Undefined: Representa “valor no asignado”.

Objetc**:**

Los objetos se utilizan para guardar conjuntos de datos y entidades complejas como:

- Object, Array, Function, Date, entre otros.
- **Symbol:** Se utiliza para crear identificadores únicos en los object

Typeof**:**

Devuelve una cadena con el nombre del tipo de dato. Ej: let a = typeof true devuelve boolean

Variables:

Una variable es un almacén con un nombre, para guardar datos, para declarar una variable en JavaScript se utiliza la palabra “Var” o “Let” seguido de un nombre de significado evidente.

Constantes:

Una constante es una variable constante, lo cual significa que no se puede modificar, para declarar una constante en JavaScript se utiliza la palabra “const” seguido de su nombre.

Conversiones:

Se llama conversiones de tipo cuando los operadores y funciones convierten automáticamente los valores que se le pasan al tipo que necesitan. Entre estos están:

- ToString: Se utiliza cuando se necesita la representación en String de un valor.
- ToNumber: Se utiliza cuando se necesita la representación en Number de un valor.
- ToBoolean: Se utiliza cuando se necesita la representación en Boolean de un valor.

Operadores aritméticos-relacionales-lógicos:

Los operadores aritméticos son:

- La suma (+): se usa para la suma de dos o más valores.
- La resta (-): se usa para la resta de dos o más valores.
- La multiplicación (*): se usa para la multiplicación de dos o más valores.
- La división (/): se usa para la división de dos o más valores.
- El Resto (%): se usa para la saber el valor del residuo de una división.
- La exponenciación (**): se usa para representar un valor con exponente.

Los operadores relacionales son:

- Igualdad débil (==): compara dos valores para verificar si son iguales en valor.
- Igualdad estricta (===): compara dos valores para verificar si son iguales en valor y en tipo de dato.
- Diferente débil (!=): verifica si dos valores no son iguales en valor.
- Diferente estricta (!==): verifica si dos valores no son iguales en valor o en tipo de dato.
- Mayor que (>): compara dos valores para verificar si el operando de la izquierda es mayor que el operando de la derecha.
- Menor que (<): compara dos valores para verificar si el operando de la izquierda es menor que el operando de la derecha.
- Mayor o igual que (>=): compara dos valores para verificar si el operando de la izquierda es mayor o igual que el operando de la derecha.
- Menor o igual que (<=): compara dos valores para verificar si el operando de la izquierda es menor o igual que el operando de la derecha.

Los operadores lógicos son:

- Y (&&): se usa como un “y”.
- O (||): se usa como un “o”.
- No (!): se usa como un “no” o para decir que es “diferente”.

Expresiones:

Las expresiones son un sistema para buscar, capturar o reemplazar texto utilizando patrones. Estos patrones permiten realizar una búsqueda de texto de una forma relativamente sencilla y abstracta, de forma que abarca una gran cantidad de posibilidades que de otra forma sería imposible o muy extensa y compleja.

Jerarquías:

La jerarquía se refiere a la estructura de organización en la que se encuentran las propiedades y métodos dentro de un objeto. En términos más simples, es la manera en la que las propiedades y métodos están agrupados y ordenados en relación con el objeto en sí y sus subobjetos.

Condicionales:

Una condicional es un método el cual nos permite tomar decisiones en nuestro código por medio de una entrada, tenemos 3 tipos de condiciones:

If:

La sentencia if(…) evalúa la condición en los paréntesis y si el resultado es true ejecuta un bloque de código.

Ejemplo:

```
let year = prompt('¿En que año fué publicada la especificación EC

if (year == 2015) alert( '¡Estás en lo cierto!' );
```

Else:

Este se ejecutará cuando la condición en el bloque sea falsa.

Por ejemplo:

```
let year = prompt('¿En qué año fue publicada la especificación EC

if (year == 2015) {
  alert( '¡Lo adivinaste, correcto!' );
} else {
  alert( '¿Cómo puedes estar tan equivocado?' ); // cualquier val
}
```

Else if:

Esta condición nos permite probar variantes en el bloque de código.

Por ejemplo:

```
let year = prompt('¿En qué año fue publicada la especificación EC

if (year < 2015) {
  alert( 'Muy poco...' );
} else if (year > 2015) {
  alert( 'Muy Tarde' );
} else {
  alert( '¡Exactamente!' );
}
```

Switch-case:

La estructura switch es una alternativa a la estructura if- else, se utiliza para evaluar una expresión y ejecutar un bloque de código dependiendo del valor de esa expresión.

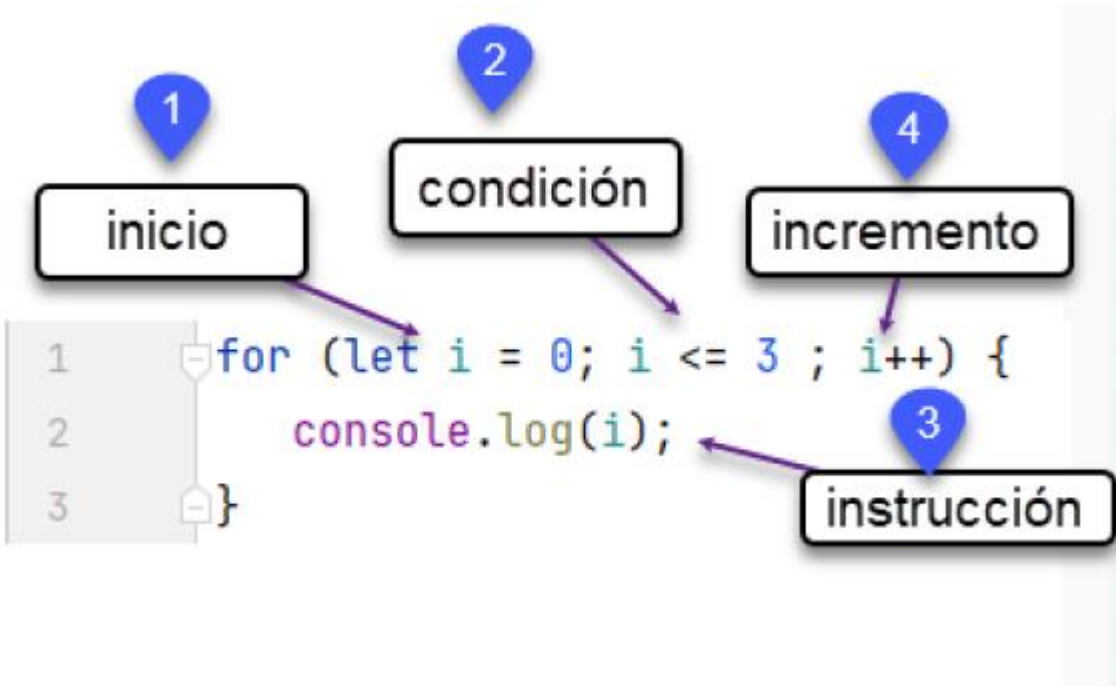
```
let a = 2 + 2;

switch (a) {
  case 3:
    alert( 'Muy pequeño' );
    break;
  case 4:
    alert( '¡Exacto!' );
    break;
  case 5:
    alert( 'Muy grande' );
    break;
  default:
    alert( "Desconozco estos valores" );
}
```

Ciclos:

Un ciclo en programación, también conocido como bucle, es una estructura que permite ejecutar repetidamente un conjunto de instrucciones o un bloque de código mientras se cumpla una condición específica.

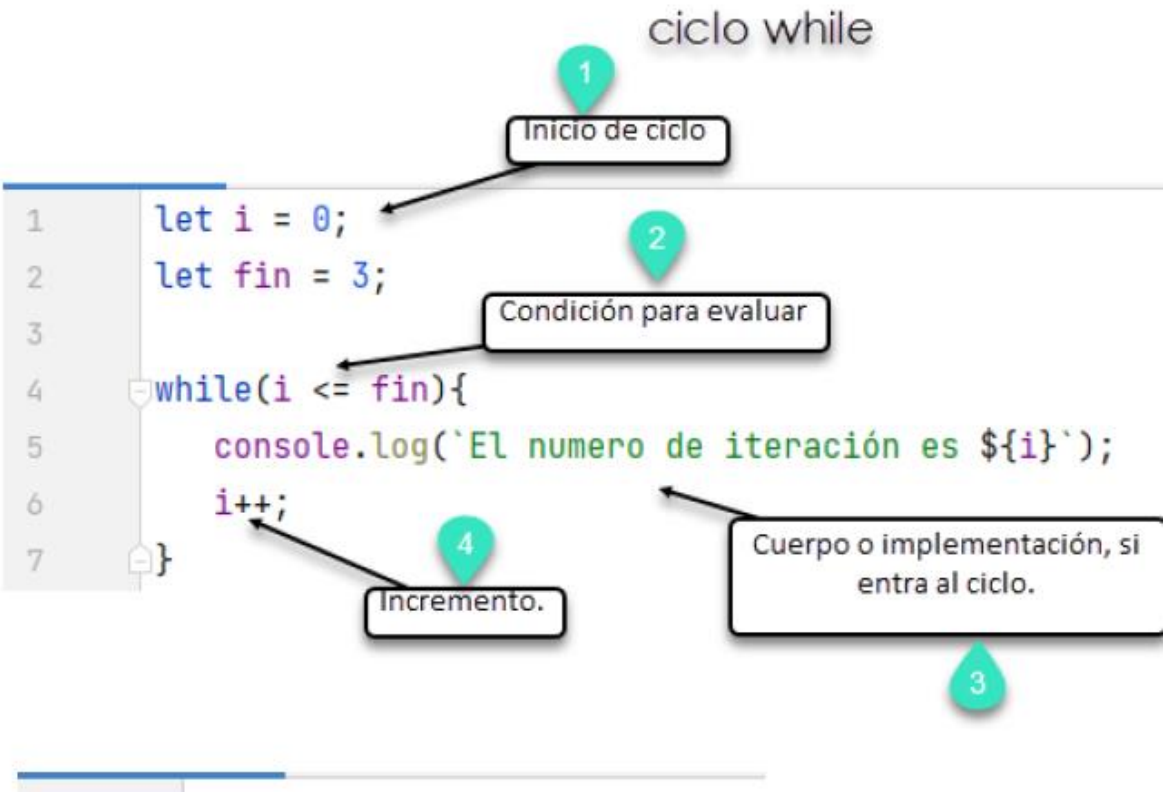
Ciclo For:

El ciclo for en JavaScript es una estructura que repite un bloque de código según tres expresiones definidas al momento de su declaración. Estas expresiones controlan la inicialización, la condición de continuación y el paso de iteración del ciclo, ofreciendo un control preciso sobre la ejecución repetida del código.



Ciclo While:

El ciclo while en JavaScript es una estructura que repite un bloque de código mientras una condición especificada sea verdadera. En otras palabras, continúa ejecutándose mientras la expresión de la condición evaluada sea true.



Ciclo Do-While:

El bucle do-while se utiliza para repetir un conjunto de instrucciones mientras la condición proporcionada se evalúe como verdadera. La principal característica distintiva del bucle do-while es que la evaluación de la condición se realiza al final de cada iteración, lo que asegura que el bloque de código contenido en el bucle se ejecute al menos una vez.