

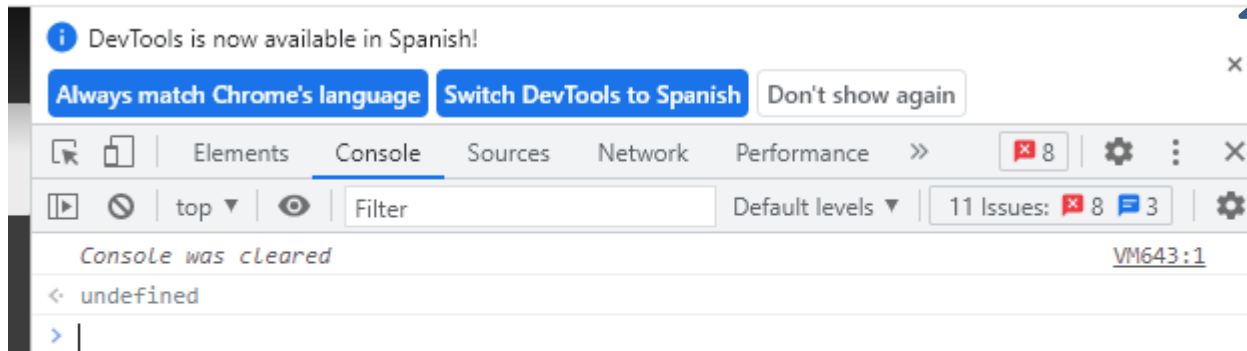
¿Qué es JavaScript?

- Lenguaje de programación encargado de dotar de mayor **interactividad y dinamismo a las páginas web.**
- Lenguaje Interpretado
- Orientado a Objetos
- Débilmente tipado y dinámico
- **No** es lo mismo que **Java**

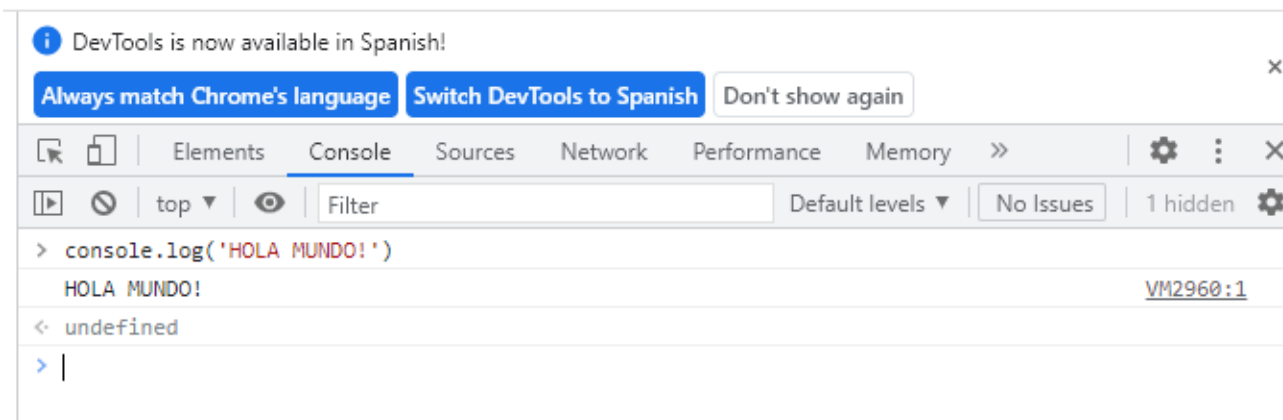


Consola de comandos

F12 o bien, *Mas herramientas>>Herramientas de desarrollo desde el navegador*



- Esta opción muestra información sobre la página que se está ejecutando
- También tiene una **consola** que permite evaluar expresiones **JS**.
- **console.log('HOLA MUNDO!')** permite mostrar mensajes en línea de comandos.



Comentarios

- ✓ En **JS** existen dos formas de escribir un comentario:



// Comentario en una sola línea

/* Comentario en más de una
línea de código

***/**

- ✓ Los comentarios son **anotaciones ignoradas** por el **intérprete**
- ✓ Permiten mejorar la **comprensión del código**

Variables

- Las variables se utilizan para almacenar los datos de nuestra aplicación en memoria (**entradas y resultados**)
- Existen tres tipos de declaraciones en **JS**:
 - ✓ **var**: permite declarar variables que pueden ser modificadas y re-declaradas dentro de su ámbito 
 - ✓ **let**: las variables pueden ser modificadas pero no re-declaradas 
 - ✓ **const**: por último las constantes no pueden ser modificadas ni re-declaradas
- El nombre de la variables recibe el nombre de **IDENTIFICADOR**

Variables

IDENTIFICADORES

- ✓ Los nombres de variables deben empezar por letra (mayúscula o minúscula), un carácter de subrayado (_) o un signo pesos (\$).
- ✓ No pueden empezar con un número
- ✓ No pueden contener puntuación, espacios o guiones (medios)
- ✓ No pueden coincidir con palabras reservadas (como `const` o `case`) del lenguaje
- ✓ Algunas convenciones:

- Para constantes **UPPER_CASE**:

```
const UNA_CONSTANTE
```

- Para tipos primitivos y objetos **LOWER CAMEL CASE**:

```
let unaCadena = "un string"
```



Tipo de datos en JS

✓ Primitivos:

- **string**: secuencia de caracteres. Por ejemplo: 'Cadena'
- **number**: valor numérico (entero o con coma): 44 o 3.15
- **Boolean**: representan un valor lógico: true/false
- **null**: denota un valor nulo asignado por el programador
- **undefined**: valor sin definir asignado automáticamente por JS
- **NaN**: No es un número: división 0/0 por ejemplo, o bien "cadena"*3

✓ Compuestos:

- **object**: es una **colección** de pares de valores **clave-valor**. Cada par clave valor se llama **propiedad**
- **array**: son conjunto de valores almacenados en una misma variable.

Cadenas en JS

✓ Ejemplos:

- `let` cadena = “una cadena de ejemplo”
- `let` otraCadena = cadena + “ concatenada”
- `let` nuevaCadena = `new` String(“Cadena como objeto”)

✓ Interpolación: (Template String/2015)

- ✓ `Let` correo = “martindapol@gmail.com”
- ✓ `let` mensaje = `Mi correo es: \${correo}`
- ✓ `console.log` (mensaje)

✓ Métodos:

- `trim()`, `toUpperCase()`, `toLowerCase()`, `includes()`, `split()`, `length`,...

Números y valores lógicos

✓ Ejemplos:

- `let verdadero = true`
- `let falso = new Boolean(false)`
- `console.log(verdadero, falso)`

✓ ¿Qué cosas son falsas para JS:

- Cadenas vacías → false
- número cero → false
- null → false
- undefined → false
- NaN → false

MDN truthy
<https://developer.mozilla.org/es/docs/Glossary/Truthy>



Números y valores lógicos

✓ Ejemplos:

- `let num1 = 5`
- `let num2 = new Number(3.14)`
- `console.log(num1, num2)`

✓ Métodos:

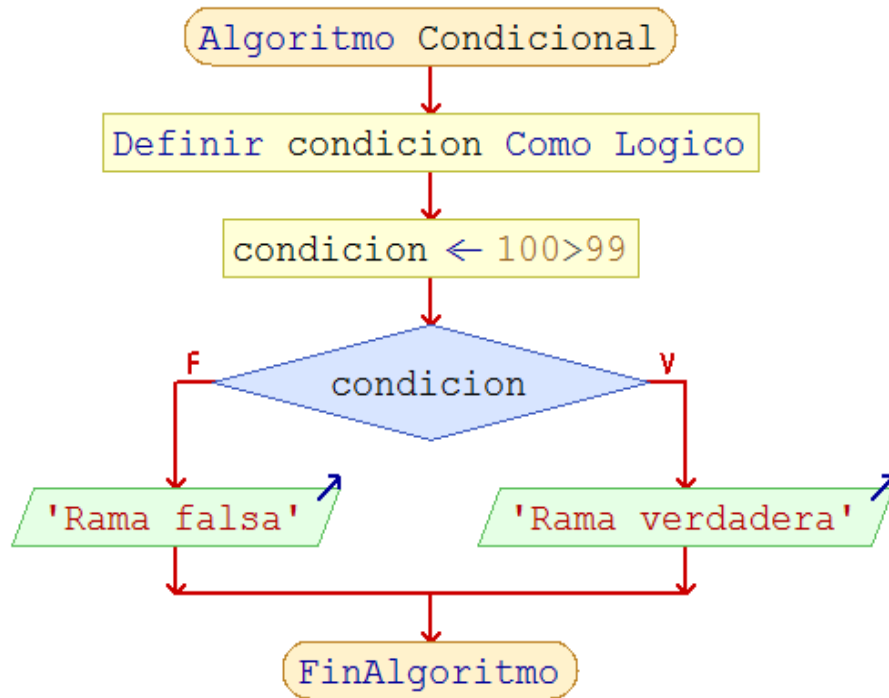
- `toFixed(decimales)`, `parseInt(val)`, `parseFloat(val)`, `typeof(var)`,



- Cuidado con **concatenar tipos diferentes:**

- `let num1 = "5"`
- `let num2 = 7`
- `console.log(num1 + num2)` → Esto imprime "75"

Estructuras condicionales



Sintaxis **JS**:

```
if (expresión lógica) {  
    instrucciones de la rama verdadera  
} else {  
    instrucciones de la rama falsa  
}
```

Estructuras condicionales



Operadores lógicos

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo
!	Negación - NOT (unario)	<code>!false</code> <code>!(5==5)</code>	true false
 	Suma lógica con cortocircuito: si el primer operando es true entonces el segundo se salta y el resultado es true	<code>true false</code> <code>(5==5) (5<4)</code>	true true
&&	Producto lógico con cortocircuito: si el primer operando es false entonces el segundo se salta y el resultado es false	<code>false && true</code> <code>(5==5) && (5<4)</code>	false false

Operadores relacionales

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo
==	igual que	<code>7 == 38</code>	false
!=	distinto que	<code>'a' != 'k'</code>	true
<	menor que	<code>'G' < 'B'</code>	false
>	mayor que	<code>'b' > 'a'</code>	true
<=	menor o igual que	<code>7.5 <= 7.38</code>	false
>=	mayor o igual que	<code>38 >= 7</code>	true

Estructuras condicionales

Operador ternario ?:

```
Let entrada = -1;
```

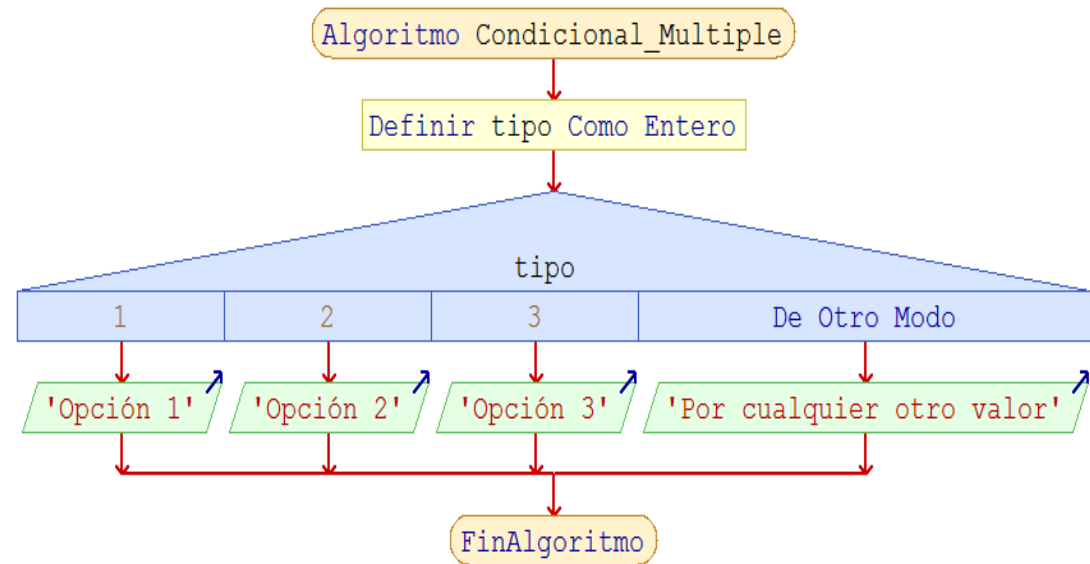
```
Let valor = entrada > 0 ?
```

```
entrada : 0;
```

```
console.log(valor)
```

Condicional múltiple:

```
switch (expresión) {  
  case (valor1) : instrucciones_1;  
  [break;]  
  case (valor2) : instrucciones_2;  
  [break;]  
  ...  
  case (valorN) : instrucciones_N;  
  [break;]  
  default: instrucciones_por_defecto;  
}
```



Estructuras condicionales

Vamos a practicar



Caso práctico 01

- Escribir un programa que convierta la hora de un día de notación de 24 horas y genera una salida como respuesta en formato 12 horas.
- Por ejemplo, si la entrada es **13:45**, la salida será: **1:45 PM**
- El programa deberá generar las salidas por consola para los siguientes casos:
 - 00:15 → 12:15 AM
 - 10:30 → 10:30 AM
 - 18:25 → 6:25 PM

Caso práctico 02

- Dado un número de tres cifras, desarrolle un programa **JavaScript** que permita invertir dicho número.

Por ejemplo: 865 → 568

- Además deberá indicar con un mensaje si el número es el mismo del derecho y el revés.
- En caso de que ingrese un valor no deseado indicar con un mensaje la situación