Tabla de contenido

[**Actividad JavaScript** 2](#_Toc109813016)

[**1.** **Declaración de variables y constantes** 2](#_Toc109813017)

[- **Variables** 2](#_Toc109813018)

[- **Constantes** 2](#_Toc109813019)

[**2.** **Operadores (aritméticos, relaciones o de operación lógicos, asignación)** 3](#_Toc109813020)

[- **Operadores aritméticos** 3](#_Toc109813021)

[- **Operadores asignación** 3](#_Toc109813022)

[- **Operadores comparación** 3](#_Toc109813023)

[- **Operadores lógicos** 4](#_Toc109813024)

[- **Operadores** **tipo** 4](#_Toc109813025)

[- **Operadores de bit a bit** 4](#_Toc109813026)

[- **Operadores de cadena** 4](#_Toc109813027)

# **Actividad JavaScript**

## **Declaración de variables y constantes**

### **Variables**

Una variable es un “almacenamiento con nombre” para los datos. Podemos usar variables para almacenar golosinas, visitantes y otros datos.

Para crear una variable en JavaScript, necesitamos usar la palabra clave let.

Paso a paso:

Let Mensaje;  
aquí le decimos a Java que nuestra variable se llama mensaje

Let Mensaje;

Mensaje = ‘hola aprendiz’;

Aquí le damos contenido a nuestra variable el cual es -hola aprendiz

Let Mensaje;

Mensaje = ‘hola aprendiz’;

Alert(Mensaje);

Muestra el contenido de la variable mensaje

Abreviado:

Let Mensaje = ‘hola aprendiz’;

Estamos asignando y declarando la variable en solo una línea de código

Let Persona = ‘Juan’, Edad = 17, Mensaje = ‘hola aprendiz’

Estamos asignando y declarando varias variables en solo una línea de código

### **Constantes**

Para declarar una variable constante (invariable), se puede usar const en lugar de let:

Const nacimiento = ’14.02.2005’

Las variables declaradas usando const se llaman “constantes”. No se pueden cambiar. Un intento de hacerlo causaría un error

Beneficio Constantes:

Existe una práctica generalizada de usar constantes como alias para valores difíciles de recordar que se conocen antes de la ejecución.

Tales constantes se nombran usando letras mayúsculas y guiones bajos.

Ejemplos:

Const COLOR\_ROJO = “# F00”;

Const COLOR\_VERDE = “# 0F0”;

Const COLOR\_AZUL = “# 00F”;

Const COLOR\_NARANJA = “# FF7F00”;

Esto lo usamos porque es más fácil recordad el nombre de la constante que recordad el numeral del color

## **Operadores (aritméticos, relaciones o de operación lógicos, asignación)**

### **Operadores aritméticos**

Los operadores aritméticos se utilizan para realizar operaciones aritméticas con números:

|  |  |
| --- | --- |
| Operadores | Descripción |
| + | Suma |
| - | Resta |
| \* | Multiplicación |
| \*\* | Exponenciación |
| / | División |
| % | Modulo (mod) |
| ++ | Incremento |
| -- | Decremento |

### **Operadores asignación**

Los operadores de asignación asignan valores a las variables de JavaScript

|  |  |  |
| --- | --- | --- |
| Operadores | Ejemplo | Igual que |
| = | x = y | x = y |
| += | x += y | x = x + y |
| -= | x -= y | x = x - y |
| \*= | x \*= y | x = x \* y |
| /= | x /= y | x =x / y |
| %= | x %= y | x = x % y |
| \*\*= | x \*\*= y | x = x \*\* y |

### **Operadores comparación**

|  |  |
| --- | --- |
| Operadores | Descripción |
| == | Igual a |
| === | Igual valor e igual tipo |
| != | No igual |
| !== | Valor diferente o tipo diferente |
| > | Mayor que |
| < | Menor que |
| >= | Mayor o igual que |
| <= | Menor o igual que |
| ? | Operador ternario |

### **Operadores lógicos**

|  |  |
| --- | --- |
| Operador | Descripción |
| && | Lógico y |
| || | Lógico o |
| ! | No lógico |

### **Operadores** **tipo**

|  |  |
| --- | --- |
| Operador | Descripción |
| typeof | Devuelve el tipo de una variable |
| instanceof | Devuelve verdadero si un objeto es una instancia de un tipo objeto |

### **Operadores de bit a bit**

Los operadores de bits funcionan en números de 32 bits.

Cualquier operando numérico en la operación se convierte en un número de 32 bits. El resultado se vuelve a convertir en un número de JavaScript.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Operador | Descripción | Ejemplo | Igual que | Resultado | Decimal |
| & | Y | 5 & 1 | 0101 & 0001 | 0001 | 1 |
| | | O | 5 | 1 | 0101 | 0001 | 0101 | 5 |
| ~ | No | ~5 | ~0101 | 1010 | 10 |
| ^ | Función desigualdad | 5 ^ 1 | 0101 ^ 0001 | 0100 | 4 |
| << | Desplazamiento a la izquierda 5 | 5 << 1 | 0101 << 1 | 1010 | 10 |
| >> | Desplazamiento a la derecha 5 | 5 >> 1 | 0101 >> 1 | 0010 | 2 |
| >>> | Desplazamiento a la derecha sin firmar | 5 >>> 1 | 0101 >>> 1 | 0010 | 2 |

### **Operadores de cadena**

El + operador también se puede usar para agregar (concatenar) cadenas.

Ejemplo:

let text1 = "John";  
let text2 = "Doe";  
let text3 = text1 + " " + text2;

resultado:

John Doe

El += operador de asignación también se puede usar para agregar (concatenar) cadenas:

Ejemplo:

let text1 = "Hola que ";  
text1 += "lindo día";

resultado:

Hola que lindo día

#### **Adición de cadenas y números**

Agregar dos números devolverá la suma, pero agregar un número y una cadena devolverá una cadena:

Ejemplo:

let x = 5 + 5;  
let y = "5" + 5;  
let z = "Hola" + 5;

resultado:

10  
55  
Hola5