


Empezamos a las
19:00hs





Expresiones y Operadores



TEMAS

- Repaso
- Variables
- var, let y const
- hoisting
- Operadores
- Precedencia de operadores
- Expresiones Regulares

Repaso de la clase anterior

- Cajas (variables)
- Listas (arrays)
- Repetición (ciclos)
- Condicional (if-else)
- Mis bloques (Funciones)

Variable

En programación, las variables son como cajas de almacenamiento donde podemos guardar diferentes cosas. Cada caja tiene un nombre y puede contener un valor específico, como números o palabras. Las variables nos permiten acceder y manipular esos valores a lo largo de nuestro programa.

El almacenamiento en la memoria RAM es volátil, es decir, al apagar la computadora o el programa la información almacenada allí se perderá

Declaración de variables

En JavaScript, podemos declarar variables utilizando diferentes palabras clave: `var`, `let` y `const`. Cada una tiene características distintas que debemos tener en cuenta al programar.

JS index.js U ●

monedas > JS index.js > ...

```
1  var edad = 25;  
2  let nombre = "Juan";  
3  const PI = 3.1416;  
4
```

Scope

En JavaScript, el scope (alcance) se refiere a la visibilidad y accesibilidad de las variables, funciones y objetos en una parte específica del código durante la ejecución del programa. Define las reglas sobre cómo se accede y se maneja la información dentro de un bloque de código.

El scope en JavaScript se basa en la estructura de los bloques de código, como las funciones y los bloques de declaración (como los bloques if, for, while, etc.). Cada vez que se crea un nuevo bloque, se crea un nuevo scope.

var

var tiene un alcance global, no importa en que bloque la defina, en cualquier lugar posterior a su definición tendré acceso al valor

```
{  
    var activo = true  
    console.log(activo)  
}  
console.log(activo) //imprime true
```


let

Si



- Alcance de bloque
- Permite cambiar su valor
- Permite redefinición en otro bloque

No



- No permite redefinición en el mismo bloque

let

```
//alcanza a bloques hijos
clima = null
{
  console.log(clima) //null
  {
    let clima = "lluvioso"
    console.log(clima) //lluvioso
    {
      let clima = "calido"
      {
        console.log(clima) //calido
      }
    }
  }
}
```

```
//defino dentro de bloque y solamente es
accedible dentro de él
{
  let activo = true
  console.log(activo)
}
console.log(activo) //error: activo is not
defined
```

const

Si



- Alcance de bloque
- Permite redefinición en otro bloque

No



- No permite cambiar valor
- No permite redefinición en el mismo bloque

const

```
//alcanza a bloques hijos
//permite redefinir mismo nombre en otro bloque
const clima = null
{
  console.log(clima) //imprime null
  {
    const clima = "lluvioso"
    console.log(clima) //imprime lluvioso
    {
      const clima = "calido"
      {
        console.log(clima) //imprime calido
      }
    }
  }
}
```

```
//no permite reasignar valores
const peso = 95
console.log(peso)
//peso = 99 //error: Assignment to
constant variable.
console.log(peso)
```

Operadores

Los operadores son elementos fundamentales en la programación, ya que nos permiten realizar diferentes tipos de cálculos y manipular datos. En JavaScript, uno de los lenguajes de programación más utilizados en desarrollo web, encontramos una amplia variedad de operadores.

JavaScript cuenta con operadores

- aritméticos
- asignación
- comparación
- lógicos

Aritméticos

Nombre	Operador abreviado	Significado
Suma	+	$a + b$
Resta	-	$a - b$
División	/	a / b
Multiplicación	*	$a * b$
Módulo	%	$a \% b$

Asignación

Nombre	Operador abreviado	Significado
Asignación	$x = y$	$x = y$
Asignación de adición	$x += y$	$x = x + y$
Asignación de resta	$x -= y$	$x = x - y$
Asignación de multiplicación	$x *= y$	$x = x * y$
Asignación de división	$x /= y$	$x = x / y$

Asignación

Nombre	Operador abreviado	Significado
Asignación de residuo	<code>x %= y</code>	<code>x = x % y</code>
Asignación de exponenciación	<code>x **= y</code>	<code>x = x ** y</code>
Asignación OR lógico	<code>x = y</code>	<code>x (x = y)</code>
Asignación AND lógico	<code>x &&= y</code>	<code>x && (x = y)</code>

Comparación

Nombre	Operador	Ejemplo
Igualdad de valores	<code>==</code>	<code>a == b</code>
Igualdad de valores y tipo	<code>===</code>	<code>a === b</code>
Mayor que	<code>></code>	<code>a > b</code>
Mayor o igual que	<code>>=</code>	<code>a >= B</code>

Comparación

Nombre	Operador	Ejemplo
Menor que	<	$a < b$
Menor o igual que	<=	$a <= b$
Distinto	!=	$a != b$
Estrictamente distinto	!==	$a !== b$

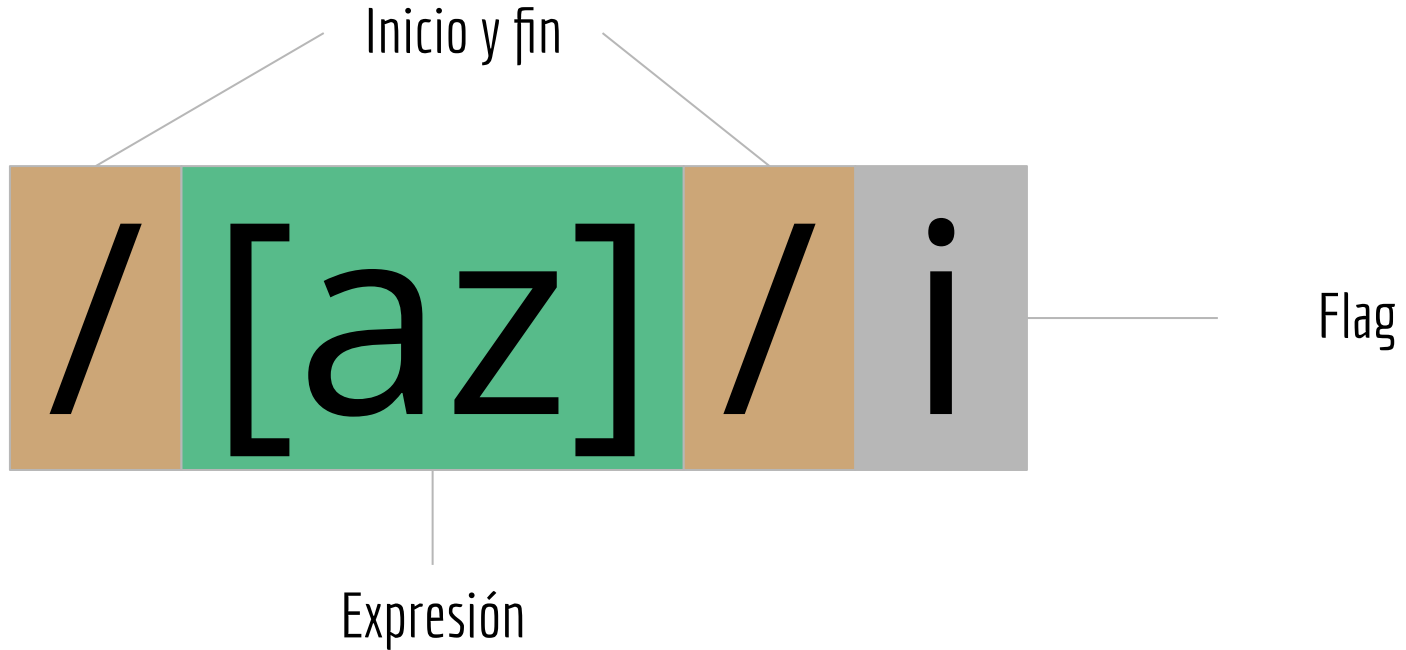
Lógicos

Nombre	Operador	Ejemplo
AND	&&	a && b
OR		a b
NOT	!	!a

Expresiones Regulares

Una expresión regular en programación es una secuencia de caracteres que define un patrón de búsqueda. Se utiliza para buscar y manipular texto de manera eficiente. Las expresiones regulares son utilizadas en muchos lenguajes de programación y aplicaciones para realizar tareas como validar formatos de entradas, buscar y reemplazar texto, y extraer información específica de cadenas de caracteres. Proporcionan una forma flexible y poderosa de trabajar con texto al permitir especificar reglas precisas sobre qué tipo de cadenas deben coincidir con el patrón definido.

Expresiones Regulares





Live code





¡Gracias!

Espero que este sea un largo
camino que transitemos juntos

