cedia | TMO Training Management Office

info@cedia.org.ec
cedia.edu.ec

CURSO DE JAVASCRIPT

Corporación Ecuatoriana para el Desarrollo de la Investigación y la Academia

Ing. María Fernanda Granda J., PhD

25-11-2025

"Conectando ideas, transformando sociedades"

Módulo 4: Alcance en JavaScript

- ¿Qué es el alcance y para qué sirve?
- Niveles de alcance.
- Alcance implícito.
- Palabras reservadas var vs. let.
- Implicaciones del nivel de alcance

© El presente documento se encuentra protegido por Derechos de Autor y es de titularidad exclusiva de CEDIA. Ha sido autorizado su uso exclusivo para fines informativos. Queda totalmente prohibida la distribución total o parcial no autorizada de los contenidos del documento para fines ajenos al previsto. Cualquier duda de aplicación individual o interpretación puede ser comunicada al correo: info@cedia.org.ec. El presente aviso no podrá ser modificado, ni excluido del texto.

¿Qué es el alcance y para qué sirve?

- Alcance (scope) de una variable define dónde y cuándo es visible y accesible.
- Determina qué variables pueden ser usadas en cada parte del código.

¿Y para qué nos sirve el Scope?

- Delimita quienes tienen acceso y quienes no a determinadas partes del código
- También facilitará la detección y disminución de errores, por ende el código será más robusto.

© El presente documento se encuentra protegido por Derechos de Autor y es de titularidad exclusiva de CEDIA. Ha sido autorizado su uso exclusivo para fines informativos. Queda totalmente prohibida la distribución total o parcial no autorizada de los contenidos del documento para fines ajenos al previsto. Cualquier duda de aplicación individual o interpretación puede ser comunicada al correo: info@cedia.org.ec. El presente aviso no podrá ser modificado, ni excluido del texto.

Niveles de Alcance

❑ Alcance Global

Si una variable se declara fuera de todas las funciones o llaves ({}), se dice que está definida en el ámbito global.

Se puede declarar usando var, let o const

```
const saludo = 'Hola estudiantes de JavaScript!'

function decirHola () {
    console.log(saludo)
}

console.log(saludo) // 'Hola estudiantes de JavaScript!'
decirHola() // 'Hola estudiantes de JavaScript!'
```


totalmente prohibida la distribución total o parcial no autoriza-
da de los contenidos del documento para fines ajenos al previsto. Cualquier duda de aplicación individual o interpretación puede ser comunicada al correo: info@cedia.org.ec. El presente aviso no podrá ser modificado, ni excluido del texto.

Niveles de Alcance

Alcance Local

Este tipo de variables van a vivir únicamente dentro de la función en donde las hayamos declarado y si intentamos accederlas fuera de ella, dichas variables no van a estar definidas.

En JavaScript, hay dos tipos de ámbito local:

- ámbito de función y
- ámbito de bloque.

© El presente documento se encuentra protegido por Derechos de Autor y es de titularidad exclusiva de CEDIA. Ha sido autorizado su uso exclusivo para fines informativos. Queda totalmente prohibida la distribución total o parcial no autorizada de los contenidos del documento para fines ajenos al previsto. Cualquier duda de aplicación individual o interpretación puede ser comunicada al correo: info@cedia.org.ec. El presente aviso no podrá ser modificado, ni excluido del texto.

Niveles de Alcance

❑ Ámbito de Función

Al declarar una variable en una función, solo se puede acceder a ella dentro de la función. No se puede acceder a ella una vez que se sale de ella.

```
function decirHola () {
  var saludo = 'Hola estudiantes de JavaScript!';
  console.log(saludo)

}
decirHola() // 'Hola estudiantes de JavaScript!'
console.log(saludo) // Error saludo is not defined
```


© El presente documento se encuentra protegido por Derechos de Autor y es de titularidad exclusiva de CEDIA. Ha sido autorizado su uso exclusivo para fines informativos. Queda totalmente prohibida la distribución total o parcial no autorizada de los contenidos del documento para fines ajenos al previsto. Cualquier duda de aplicación individual o interpretación puede ser comunicada al correo: info@cedia.org.ec. El presente aviso no podrá ser modificado, ni excluido del texto.

Niveles de Alcance

❑ Ámbito de Bloque

Desde ECMAScript 6 contamos con los keyword **let** y **const** los cuales nos permiten tener un scope de bloque, esto quiere decir que las variables solo van a existir dentro del bloque de código correspondiente ({}).

Es un subconjunto del ámbito de función, ya que las funciones deben declararse con llaves (a menos que estés usando funciones de flecha con un retorno implícito).

© El presente documento se encuentra protegido por Derechos de Autor y es de titularidad exclusiva de CEDIA. Ha sido autorizado su uso exclusivo para fines informativos. Queda totalmente prohibida la distribución total o parcial no autorizada de los contenidos del documento para fines ajenos al previsto. Cualquier duda de aplicación individual o interpretación puede ser comunicada al correo: info@cedia.org.ec. El presente aviso no podrá ser modificado, ni excluido del texto.

Niveles de Alcance

❑ Alcance de Léxico

Significa que en un grupo anidado de funciones, las funciones internas tienen acceso a las variables y otros recursos de su ámbito padre. Esto significa que las funciones hijas están vinculadas lexicalmente al contexto de ejecución de sus padres.

```
function myFunction() {

    var nombre = 'Juan';
    // no podemos acceder a preferencias desde acá

    function parent() {

        // nombre si es accesible desde acá.
        // preferencias en cambio, no es accesible.

        function child() {

            // tambien podemos acceder a nombre desde acá.
            var preferencias = 'Codear';
        }
    }
}
```


© El presente documento se encuentra protegido por Derecho
da de los contenidos del documento para fines ajenos al prev

Alcance Implícito

❑ Global automática

Si asignamos un valor a una variable que no ha sido declarada, esta se convertirá automáticamente en una variable global.

```
saludo = 'Hola estudiantes de JavaScript!';

function decirHola() {
    console.log(saludo);
}
decirHola(); // 'Hola estudiantes de JavaScript!';
console.log(saludo); // 'Hola estudiantes de JavaScript!';
```


© El presente documento se encuentra protegido por Derechos de Autor y es de titularidad exclusiva de CEDIA. Ha sido autorizado su uso exclusivo para fines informativos. Queda totalmente prohibida la distribución total o parcial no autorizada de los contenidos del documento para fines ajenos al previsto. Cualquier duda de aplicación individual o interpretación puede ser comunicada al correo: info@cedia.org.ec. El presente aviso no podrá ser modificado, ni excluido del texto.

Palabras reservada Var vs Let

- Utilizar siempre 'let' para variables que puedan ser modificadas y reasignadas en un futuro
- 'const' para variables que no se vayan a reasignar
- Evitar utilizar 'var'.

	VAR	LET	CONST
Declaración sin valor	✅ SI	✅ SI	❌ NO
Scope	función	bloque	bloque
Reasignación	✅ SI	✅ SI	❌ NO
Redeclaración	✅ SI	❌ NO	❌ NO

© El presente documento se encuentra protegido por Derechos de Autor y es de titularidad exclusiva de CEDIA. Ha sido autorizado su uso exclusivo para fines informativos. Queda totalmente prohibida la distribución total o parcial no autorizada de los contenidos del documento para fines ajenos al previsto. Cualquier duda de aplicación individual o interpretación puede ser comunicada al correo: info@cedia.org.ec. El presente aviso no podrá ser modificado, ni excluido del texto.

Implicaciones del nivel de alcance

Existe la posibilidad de conflictos de nombres, donde dos o más variables tienen el mismo nombre.

Si se declaran las variables con **const** o **let**, se recibirá un error cada vez que se produzca un conflicto de nombres.

Si declaras tus variables con **var**, la segunda variable sobrescribe la primera después de declararla. Esto tampoco es recomendable, ya que dificulta la depuración del código.

```
var saludo = 'Hola estudiantes de JavaScript!';
var saludo = 'Esto sobreescribirá el saludo';
console.log(saludo); // 'Esto sobreescribirá el saludo';
```


© El presente documento se encuentra protegido por Derechos de Autor y es de titularidad exclusiva de CEDIA. Ha sido autorizado su uso exclusivo para fines informativos. Queda totalmente prohibida la distribución total o parcial no autorizada de los contenidos del documento para fines ajenos al previsto. Cualquier duda de aplicación individual o interpretación puede ser comunicada al correo: info@cedia.org.ec. El presente aviso no podrá ser modificado, ni excluido del texto.

Crear multiples variables

Dos maneras de crear múltiples variables:

1. **Sólo crearlas en una sola línea**

```
let numero, numero2, numero3;

numero = 2;
numero2 = 9;
numero3 = 13
```

2. **Crearlas e inicializarlas en una sóla línea**

```
let numero1 = 23, numero2 = 39
```


© El presente documento se encuentra protegido por Derechos de Autor y es de titularidad exclusiva de CEDIA. Ha sido autorizado su uso exclusivo para fines informativos. Queda totalmente prohibida la distribución total o parcial no autorizada de los contenidos del documento para fines ajenos al previsto. Cualquier duda de aplicación individual o interpretación puede ser comunicada al correo: info@cedia.org.ec. El presente aviso no podrá ser modificado, ni excluido del texto.

Operadores de Asignación

Nombres	Abreviaciones	Significado
Asignación	`x = y`	`x = y`
Asignación de adicción	`x += y`	`x = x + y`
Asignación de sustracción	`x -= y`	`x = x - y`
Asignación de multiplicación	`x *= y`	`x = x * y`
Asignación de división	`x /= y`	`x = x / y`
Asignación de Resto	`x %= y`	`x = x % y`
Asignación de exponenciación	`x **= y`	`x = x ** y`
Asignación de desplazamiento a la izquierda	`x <<= y`	`x = x << y`
Asignación de desplazamiento a la derecha	`x >>= y`	`x = x >> y`
Asignación sin signo de desplazamiento a la derecha	`x >>>= y`	`x = x >>> y`
Asignacion AND	`x &= y`	`x = x & y`
Asignacion XOR	`x ^= y`	`x = x ^ y`
Asignacion OR	`x |= y`	`x = x | y`

ente prohibida la distribución total o parcial no autoriza-
nte aviso no podrá ser modificado, ni excluido del texto.

Operadores Aritméticos

Operadores aritméticos binarios:

- **Suma (+):** `a + b` suma los valores de `a` y `b`.
- **Resta (-):** `a - b` resta el valor de `b` de `a`.
- Multiplicación ():*: `a * b` multiplica los valores de `a` y `b`.
- **División (/):** `a / b` divide el valor de `a` por `b`.
- **Módulo (%):** `a % b` devuelve el residuo de la división de `a` por `b`.
- **Exponenciación ()** :** `a ** b` eleva `a` a la potencia de `b`.

Operadores aritméticos unarios:

- **Incremento (++)**: `a++` o `++a` incrementa el valor de `a` en 1.
- **Decremento (--)**: `a--` o `--a` decrementa el valor de `a` en 1.
- **Negación (-)**: `-a` cambia el signo de `a` (positivo a negativo o viceversa).

ohibida la distribución total o parcial no autoriza-
o no podrá ser modificado, ni excluido del texto.

Plantillas literales (plantillas de cadenas)

Son cadenas literales que habilitan el uso de expresiones incrustadas.

Se delimitan con el caracter de comillas o tildes invertidas (` `) (acento grave), en lugar de las comillas sencillas o dobles.

Las plantillas de cadena de caracteres pueden contener marcadores, ${expresión}.

```
number="cinco";
alert(`El "${number}" =  5`);
```


© El presente documento se encuentra protegido por Derechos de Autor y es de titularidad exclusiva de CEDIA. Ha sido autorizado su uso exclusivo para fines informativos. Queda totalmente prohibida la distribución total o parcial no autorizada de los contenidos del documento para fines ajenos al previsto. Cualquier duda de aplicación individual o interpretación puede ser comunicada al correo: info@cedia.org.ec. El presente aviso no podrá ser modificado, ni excluido del texto.

Caracteres de Escape

Por lo general devuelve un solo valor () o ${}

Dentro de los backticks se puede usar comillas simples, dobles.

```
number="cinco";
alert(`El "${number}" = '${(3+2)}' es el resultado de sumar (3+2)`);
```


© El presente documento se encuentra protegido por Derechos de Autor y es de titularidad exclusiva de CEDIA. Ha sido autorizado su uso exclusivo para fines informativos. Queda totalmente prohibida la distribución total o parcial no autorizada de los contenidos del documento para fines ajenos al previsto. Cualquier duda de aplicación individual o interpretación puede ser comunicada al correo: info@cedia.org.ec. El presente aviso no podrá ser modificado, ni excluido del texto.

Caracteres de Escape

- Si se tiene un texto encerrado por doble comillas, se puede usar comillas simples o backsticks en el interior.

```
frase="Mi nombre es 'Ma. Fernanda Granda' instructora de CEDIA ";
alert(frase);
```

```
frase="Mi nombre es `Ma. Fernanda Granda` instructora de CEDIA ";
alert(frase);
```

- También si se tiene comilla simple al inicio

```
frase='Mi nombre es "Ma. Fernanda Granda" instructora de CEDIA ';
alert(frase);
```

```
frase='Mi nombre es `Ma. Fernanda Granda` instructora de CEDIA ';
alert(frase);
```


© El presente documento se encuentra protegido por Derechos de Autor y es de titularidad exclusiva de CEDIA. Ha sido autorizado su uso exclusivo para fines informativos. Queda totalmente prohibida la distribución total o parcial no autorizada de los contenidos del documento para fines ajenos al previsto. Cualquier duda de aplicación individual o interpretación puede ser comunicada al correo: info@cedia.org.ec. El presente aviso no podrá ser modificado, ni excluido del texto.

Caracteres de Escape

- Algunos ejemplos más:
  - `\n`: Salto de línea (new line).
  - `\r`: Retorno de carro (carriage return).
  - `\t`: Tabulación horizontal.
  - `\b`: Retroceso (backspace), utilizado en expresiones regulares.
  - `\'`: Comilla simple o apostrofe.
  - `\"`: Comilla doble.
  - `\\`: Barra invertida.
  - `\0`: Carácter nulo (null character).
  - `\u{...}`: Carácter Unicode (por ejemplo, `\u00F1` para la letra "ñ").
  - `\x{...}`: Carácter hexadecimal (por ejemplo, `\x20` para el espacio).

informativos. Queda totalmente prohibida la distribución total o parcial no autoriza-
info@cedia.org.ec. El presente aviso no podrá ser modificado, ni excluido del texto.

Operadores de Comparación

Operador	Nombre	Ejemplo	Resultado
==	Igualdad	$a == $b	Cierto si $a es igual a $b
===	Identidad	$a== =$b	Cierto si $a es igual a $b y si además son del mismo tipo (sólo PHP4 o mayor)
!=	Desigualdad	$a != $b	Cierto si $a no es igual a $b
<	Menor que	$a < $b	Cierto si $a es estrictamente menor que el de $b
>	Mayor que	$a > $b	Cierto si $a es estrictamente mayor que $b
<=	Menor o igual que	$a <= $b	Cierto si $a es menor o igual que $b
>=	Mayor o igual que	$a >= $b	Cierto si $a mayor o igual que $b

stribución total o parcial no autoriza-
er modificado, ni excluido del texto.

Ejemplo de Operadores de Comparación

```
let num1=12;
let num2=24;
alert(num1<num2 || num1==num2);
```

```
let num1=12;
let num2=24;
let num3=25;
let num4=92;
let num5=91;

let op=(num1<num2 || num2<num3) && (!(num1!=num2)&&num5!=num3);
alert(op);
```


El presente documento se encuentra protegido por derechos de autor y es de propiedad exclusiva de cedia. Su distribución total o parcial no autorizada de los contenidos del documento para fines ajenos al previsto. Cualquier duda de aplicación individual o interpretación puede ser comunicada al correo: info@cedia.org.ec. El presente aviso no podrá ser modificado, ni excluido del texto.

cedia | TMO Training Management Office

GRACIAS

cedia.edu.ec
info@cedia.org.ec
(+593) 7-407-9300