

➤ Bases del lenguaje Javascript

Plan formativo: Desarrollo de Aplicaciones Full Stack Java Trainee V2.0

HOJA DE RUTA

¿Cuáles **skill** conforman el programa?



REPASO CLASE ANTERIOR

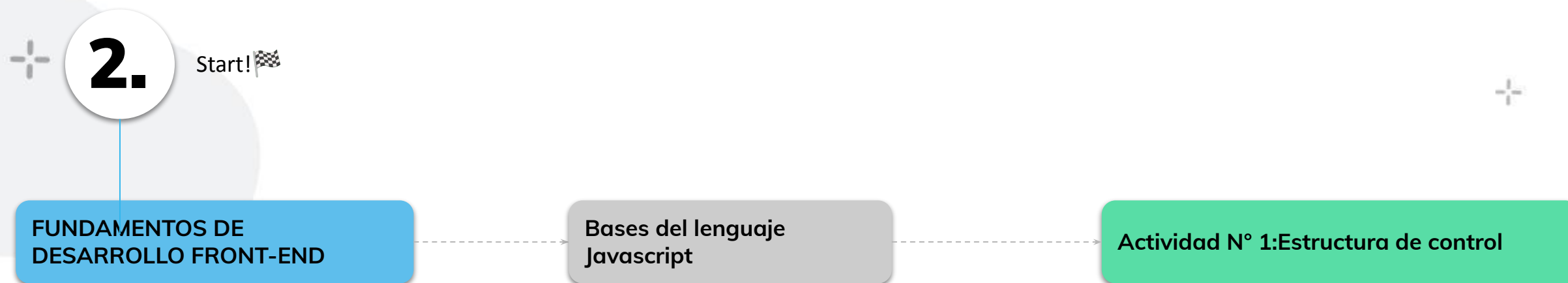
En la clase anterior trabajamos :

- ✓ Aprendimos que JavaScript se llamaba "LiveScript" pero se renombró a "JavaScript" para aprovechar la popularidad del lenguaje Java en ese momento.
- ✓ Comprender la importancia de JavaScript en el desarrollo web moderno.
- ✓ Explorar el concepto de variables en JavaScript y cómo se utilizan para almacenar y manipular datos.



LEARNING PATHWAY

Sobre qué temas trabajaremos?



En esta lección, exploraremos dos conceptos importantes en programación y diseño web. Aprenderemos cómo trabajar con expresiones aritméticas en JavaScript y aprenderemos sobre las estructuras if, else if y else y cómo se utilizan para ejecutar diferentes bloques de código según ciertas condiciones.

OBJETIVOS DE APRENDIZAJE

¿Qué aprenderemos?



Aprenderemos cómo trabajar con expresiones aritméticas en JavaScript.



Descubriremos cómo utilizar sentencias condicionales en JavaScript para tomar decisiones en nuestro código.



Aprenderemos sobre las estructuras if, else if y else y cómo se utilizan para ejecutar diferentes bloques de código según ciertas condiciones.



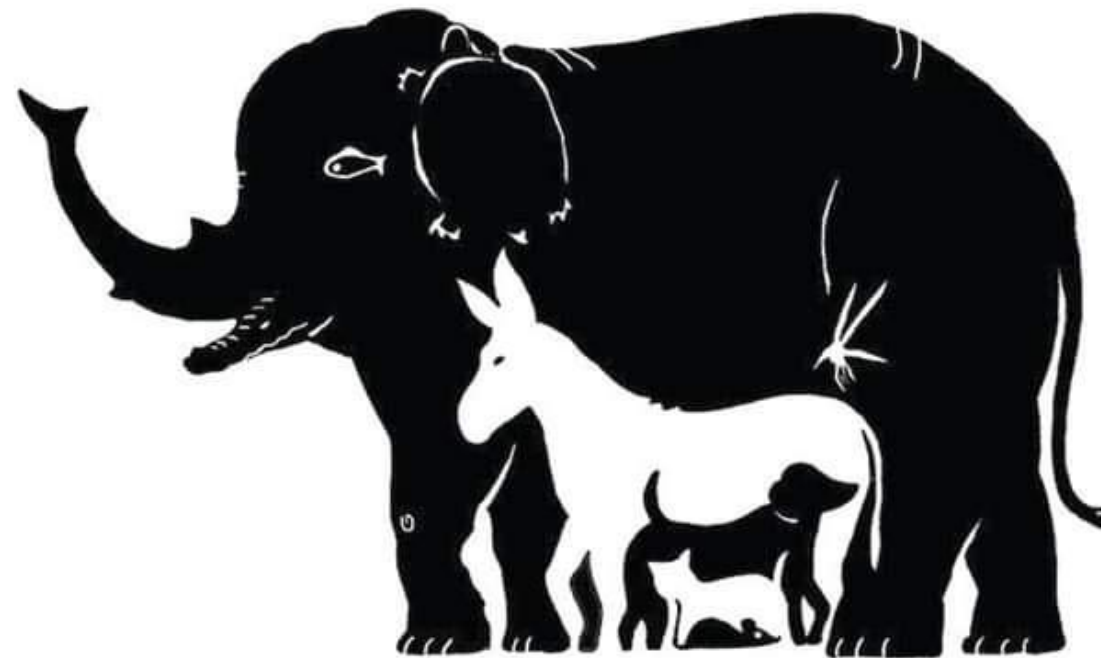


Rompehielo 🧊

Consigna: 🖋️

¿Cuántos animales ves?

**¿Estás listo para el reto del día?
Veamos...**



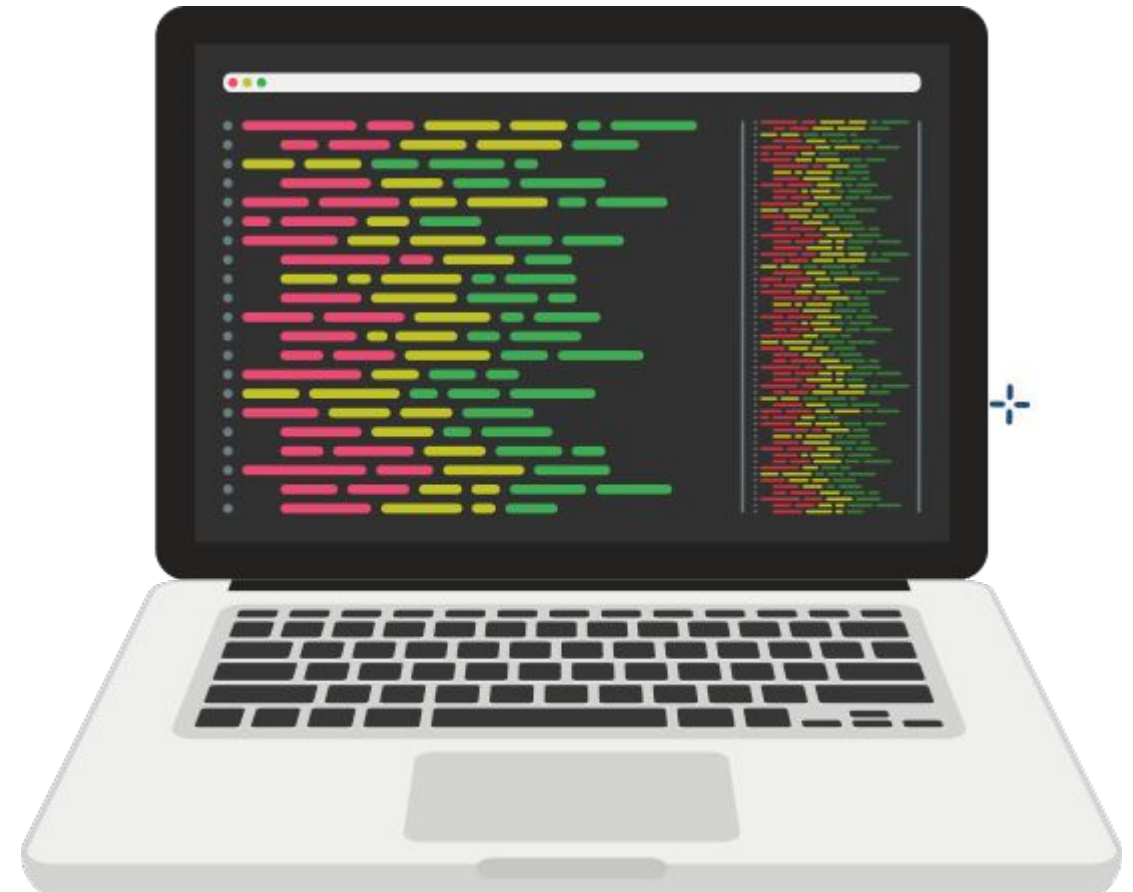
➤ Expresiones aritméticas



Expresiones aritméticas

En JavaScript, las expresiones aritméticas se utilizan para realizar operaciones matemáticas y calcular resultados numéricos.

Las expresiones aritméticas pueden involucrar operadores matemáticos como suma (+), resta (-), multiplicación (*), división (/), resto (%) y exponente (**).





Expresiones aritméticas

Los operadores aritméticos básicos, JavaScript también proporciona operadores de asignación compuestos que permiten combinar una operación aritmética con una asignación en una sola expresión.

Se utilizan diferentes operadores aritméticos para realizar operaciones matemáticas básicas. Es importante tener en cuenta que las operaciones se realizan de acuerdo con las reglas de precedencia matemática estándar.



```
let x = 5;
let y = 3;
let suma = x + y; // Suma: 8
let resta = x - y; // Resta: 2
let multiplicacion = x * y;
// Multiplicación: 15
let division = x / y;
// División: 1.6666666666666667
let resto = x % y; // Resto: 2
let exponente = x ** y; // Exponente: 125
```



Expresiones aritméticas



Suma (+)

La operación suma se realiza concatenando los números que queremos sumar con el signo de la suma (+).

```
let numero1 = 10;  
let numero2 = 50;  
console.log(10 + 50);  
console.log(numero1 + numero2)  
  
let resultadoSuma = numero1 + numero2;  
console.log(resultadoSuma)
```





Expresiones aritméticas



Resta (-)

De manera análoga, la operación resta se realiza concatenando los números que queremos restar con el signo de la resta (-).

```
let numero1 = 100;  
let numero2 = 50;  
console.log(100 - 50);  
console.log(numero1 - numero2)  
  
let resultadoResta = numero1 - numero2;  
console.log(resultadoResta)
```





Expresiones aritméticas



Multiplicación (*)

La operación multiplicación se realiza concatenando los números que queremos multiplicar con el signo de la multiplicación, el asterisco (*).

```
let numero1 = 70;  
let numero2 = 70;  
console.log(70 * 7);  
console.log(numero1 * numero2)  
  
let resultadoMultiplicacion = numero1 *  
numero2;  
console.log(resultadoMultiplicacion)
```






Expresiones aritméticas

División (/)

La operación división se realiza ubicando signo de la división, la barra inclinada hacia la derecha (/), entre los números que queremos dividir.



```
let numero1 = 80;
let numero2 = 4;
console.log(80 / 4);
console.log(numero1 / numero2)

let resultadoDivision = numero1 /
numero2;
console.log(resultadoDivision)
```





Expresiones aritméticas



% (módulo):

Calcula el resto de la división entre x e y. En el ejemplo, $x \% y$ da como resultado 2, que es el resto de la división de 5 entre 3.

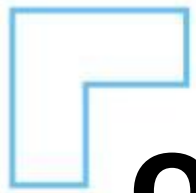
** (exponente):

Eleva x a la potencia y. En el ejemplo, $x ** y$ da como resultado 125, que es 5 elevado a la potencia 3.

```
let resto = x % y;  
// Resto: 2  
  
let exponente = x ** y;  
// Exponente: 125
```



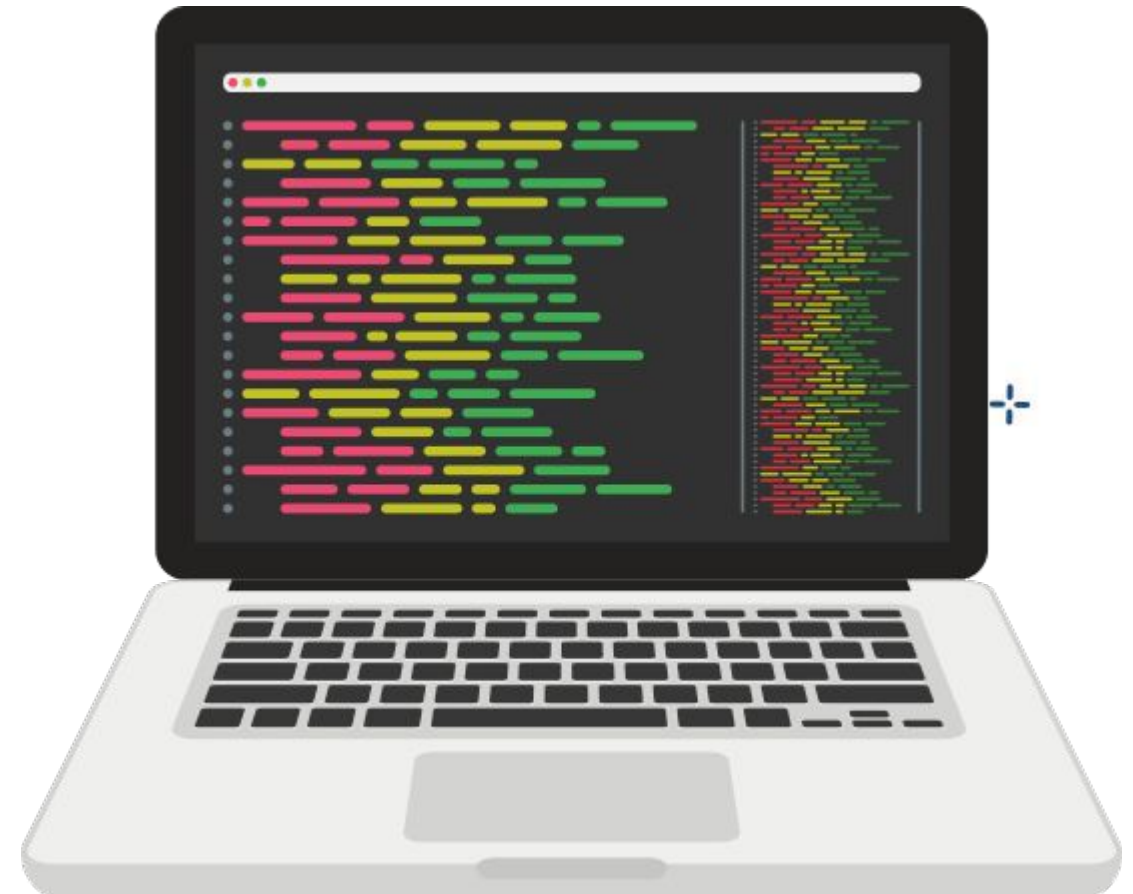
➤ Operadores de incremento y decremento



Operadores de incremento y decremento

Los operadores de incremento y decremento son herramientas poderosas para aumentar o disminuir el valor de una variable numérica de manera eficiente.

Estos operadores son útiles en situaciones donde necesitas realizar un seguimiento de valores que cambian con el tiempo, como contadores o bucles.





Operadores de incremento y decremento



Existen otros dos operadores aritméticos que aportan la utilidad de sumar o restar 1 unidad a una variable.

Cuando queremos aumentar o disminuir el valor numérico de una variable es común realizarlo utilizando el simbolo de suma “más” (+) o el de resta “menos” (-)



```
let numero = 100;
console.log(numero)
//Salida por consola --> 100
//incremento
numero = numero + 1;
console.log(numero)
//Salida por consola --> 101
//decremento
numero = numero - 1;
console.log(numero)
//Salida por consola --> 100
```



Operadores de incremento y decremento



Existe un método abreviado de hacer esto. El signo que realiza la adición en la suma se escribe ++ (con dos signos más seguidos) y en el caso la resta se escribe -- (dos signos menos seguidos).

Estos operadores (++ y --) son atajos útiles que simplifican el incremento y decremento de variables en JavaScript, haciendo que el código sea más claro y fácil de entender.

Estos dos operadores son de gran utilidad con el uso de bucles.



```
let numero = 100;
console.log(numero)
//Salida por consola --> 100
//incremento
numero++;
console.log(numero)
//Salida por consola --> 101
//decremento
numero = numero--;
console.log(numero)
//Salida por consola --> 100
```

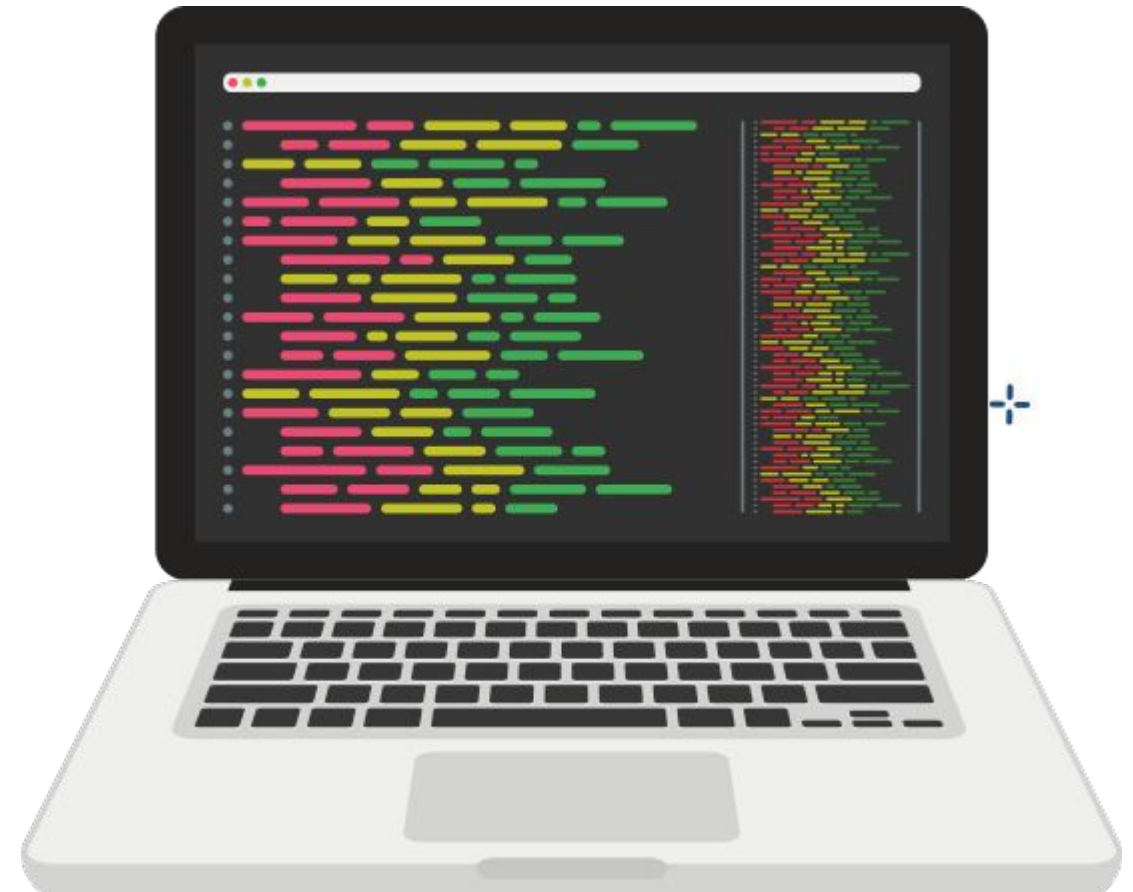
➤ Operadores de Asignación



Operadores de Asignación

Los operadores de asignación en JavaScript son utilizados para asignar valores a variables.

Estos operadores permiten no solo asignar valores simples, sino también realizar operaciones en variables mientras se les asigna un nuevo valor.





Operadores de Asignación



Los operadores de asignación en JavaScript son utilizados para asignar valores a variables. Estos operadores permiten no solo asignar valores simples, sino también realizar operaciones en variables mientras se les asigna un nuevo valor.

Se utilizan operadores de asignación compuestos junto con los operadores aritméticos básicos para actualizar el valor de la variable realizando la operación aritmética correspondiente y asignando el resultado a la misma variable.

```
let z = 10;  
z += 5; // Equivale a z = z + 5; (z = 15)  
z -= 3; // Equivale a z = z - 3; (z = 12)  
z *= 2; // Equivale a z = z * 2; (z = 24)  
z /= 4; // Equivale a z = z / 4; (z = 6)  
z %= 2; // Equivale a z = z % 2; (z = 0)
```



➤ Operadores de comparación



Operadores de comparación

Los operadores de comparación en JavaScript se utilizan para comparar dos valores y devolver un resultado booleano (true o false) que indica si la comparación es verdadera o falsa.





Operadores de comparación



Igualdad (==):

Compara si dos valores son iguales, sin tener en cuenta el tipo de dato.

Devuelve true si son iguales y false si no lo son.

Ambos valores son considerados iguales, a pesar de que uno es un número y el otro es una cadena de texto.

```
let x = 5;  
let y = "5";  
  
console.log(x == y); // Devuelve true
```





Operadores de comparación



Igualdad estricta (===):

Compara si dos valores son iguales, teniendo en cuenta tanto el valor como el tipo de dato. Devuelve true si son iguales en valor y tipo, y false si no lo son.

La comparación `x === y` devuelve false porque los valores son iguales, pero los tipos de dato son diferentes.

```
let x = 5;  
let y = "5";  
  
console.log(x === y); // Devuelve false
```





Operadores de comparación



Desigualdad (!= o !==):

Compara si dos valores no son iguales. Devuelve true si son diferentes y false si son iguales.

El operador !== también verifica el tipo de dato.

Tanto la comparación `x != y` como `x !== y` devuelven true porque los valores son diferentes.

```
let x = 5;  
let y = 10;  
  
console.log(x != y); // Devuelve true  
console.log(x !== y); // Devuelve true
```





Operadores de comparación



Otros operadores de comparación:

Además de la igualdad y desigualdad, existen otros operadores de comparación como `>`, `<`, `>=` y `<=`, que comparan si un valor es mayor, menor, mayor o igual, o menor o igual, respectivamente.

```
let x = 5;
let y = 10;

console.log(x > y); // Devuelve false
console.log(x < y); // Devuelve true
console.log(x >= y); // Devuelve false
console.log(x <= y); // Devuelve true
```



› Operadores Lógicos



Operadores Lógicos

Los operadores lógicos en JavaScript son herramientas esenciales para evaluar condiciones y tomar decisiones en la programación.

Estos operadores permiten combinar expresiones lógicas para determinar si una declaración es verdadera o falsa.

Permiten combinar expresiones y realizar operaciones lógicas en JavaScript. puedes utilizarlos para tomar decisiones basadas en múltiples condiciones o para invertir el valor de una expresión booleana.





Operadores Lógicos



Operador AND (&&):

El operador AND devuelve true si ambos operandos son true, y devuelve false en cualquier otro caso.

El operador **&&** se utiliza para evaluar dos expresiones lógicas y devuelve true si ambas expresiones son verdaderas.

```
let x = 5;
let y = 10;

console.log(x > 0 && y > 0);
// Devuelve true
console.log(x > 0 && y < 0);
// Devuelve false
```





Operadores Lógicos

Operador OR (||):

El operador OR devuelve true si al menos uno de los operandos es true, y devuelve false solo si ambos operandos son false.

El operador **||** se utiliza para evaluar dos expresiones lógicas y devuelve true si al menos una de las expresiones es verdadera.

```
let x = 5;
let y = 10;

console.log(x > 0 || y > 0);
// Devuelve true
console.log(x < 0 || y < 0);
// Devuelve false
```





Operadores Lógicos



Operador NOT (!):

El operador NOT invierte el valor de una expresión booleana. Si la expresión es true, el operador NOT devuelve false, y si la expresión es false, el operador NOT devuelve true.

El operador **!** se utiliza para negar una expresión lógica. Convierte true en false y viceversa.

```
let x = 5;
let y = 10;

console.log(!(x > 0));
// Devuelve false
console.log(!(y < 0));
// Devuelve true
```



› Sentencias condicionales

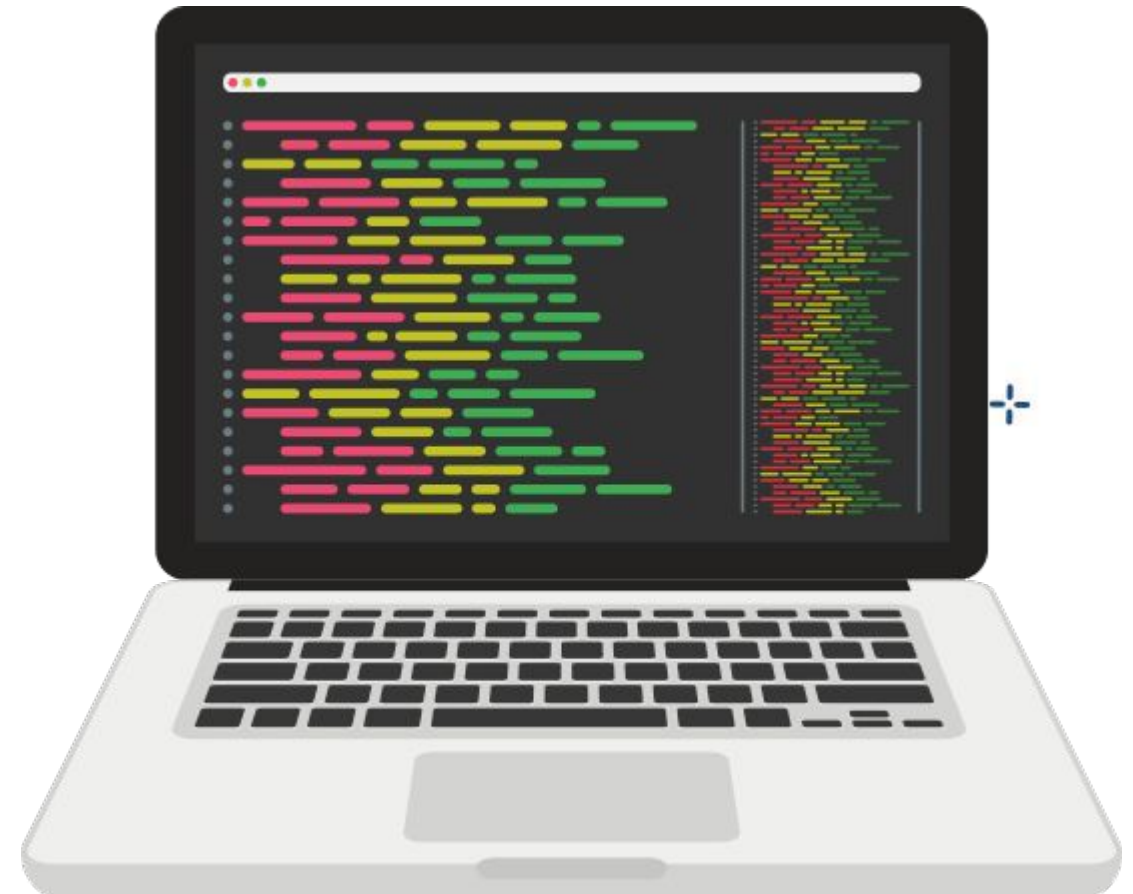


Sentencias condicionales

Las sentencias condicionales en JavaScript nos permiten tomar decisiones basadas en una condición específica. Dependiendo del resultado de esa condición, se ejecutará un bloque de código u otro.

IF - ELSE:

Es un tipo de estructura que existe en muchos lenguajes de programación y permite que el programa resuelva por su cuenta preguntas o decisiones para continuar con el flujo de tareas del usuario. Es una condicional.





Sentencias condicionales



Sentencia if():

La sentencia if se utiliza para ejecutar un bloque de código si la condición especificada es evaluada como verdadera (true).

La condición puede ser cualquier expresión que se pueda evaluar como verdadera o falsa. Si la condición es verdadera, el bloque de código dentro de las llaves {} se ejecutará. Si la condición es falsa, ese bloque se omitirá y el programa continuará con el código que sigue después del bloque if.



```
if (condición) {  
  // Este bloque de código se ejecutará si la condición es  
  verdadera.  
}  
  
let edad = 18;  
if (edad >= 18) {  
  console.log("Eres mayor de edad");  
}
```



Sentencias condicionales



Sentencia if-else:

La sentencia if-else es una estructura de control en programación que permite tomar decisiones basadas en una condición. A diferencia de la sentencia if, que ejecuta un bloque de código si una condición es verdadera y luego continúa con el resto del programa, la sentencia if-else permite ejecutar un bloque de código si la condición es verdadera y otro bloque de código si la condición es falsa.



```
if (condición) {  
  // Este bloque de código se ejecutará si la condición es  
  verdadera.  
} else {  
  // Este bloque de código se ejecutará si la condición es  
  falsa.  
}  
  
let hora = 13;  
if (hora < 12) {  
  console.log("Buenos días");  
} else {  
  console.log("Buenas tardes");  
}
```



Sentencias condicionales



Sentencia if-else if-else:

La sentencia if-else if-else nos permite evaluar múltiples condiciones y ejecutar diferentes bloques de código en función de esas condiciones.

```
let puntaje = 80;

if (puntaje >= 90) {
  console.log("Excelente");
} else if (puntaje >= 70) {
  console.log("Aprobado");
} else {
  console.log("Reprobado");
}
```





Sentencias condicionales



Operador AND (&&) con IF/ELSE:

El operador && devuelve true si ambas condiciones son verdaderas. Si alguna de las condiciones es falsa, devuelve false.

La condición dentro del if utiliza el operador && para verificar si la edad es mayor o igual a 18 y si tieneLicencia es true. Si ambas condiciones son verdaderas, se muestra el mensaje "puedes conducir legalmente". Si alguna de las condiciones es falsa, se muestra el mensaje "No cumples con los requisitos para conducir".

```
let edad = 25;
let tieneLicencia = true;

if (edad >= 18 && tieneLicencia) {
  console.log("puedes conducir legalmente.");
} else {
  console.log("No cumples con los requisitos para conducir.");
}
```



Sentencias condicionales



Operador OR (||) con IF/ELSE:

El operador || devuelve true si al menos una de las condiciones es verdadera. Solo devuelve false si ambas condiciones son falsas.

La condición dentro del if utiliza el operador || para verificar si esEstudiante es true o si esEmpleado es true. Si al menos una de las condiciones es verdadera, se muestra el mensaje "Tenes acceso a descuentos especiales". Si ambas condiciones son falsas, se muestra el mensaje "No tenes acceso a descuentos especiales".

```
let esEstudiante = false;
let esEmpleado = true;

if (esEstudiante || esEmpleado) {
  console.log("Tenes acceso a descuentos especiales.");
} else {
  console.log("No tenes acceso a descuentos especiales.");
}
```

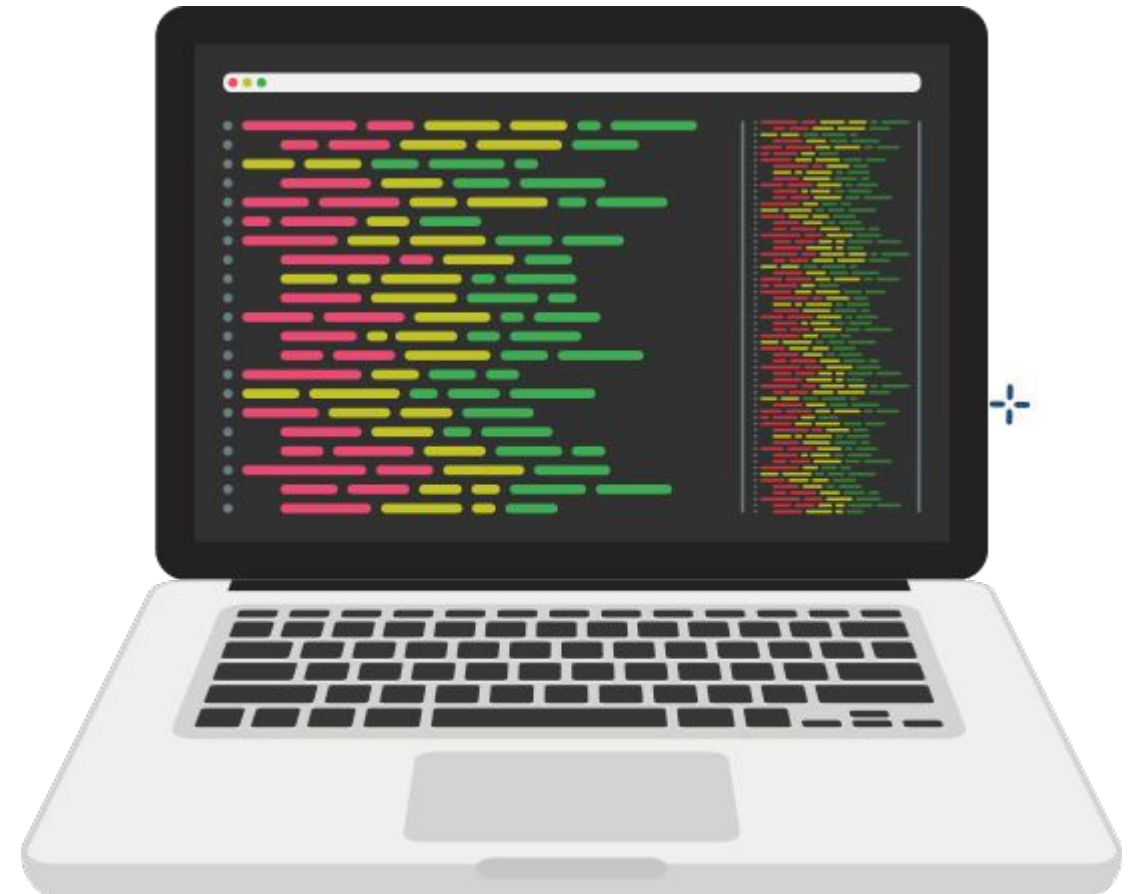
› Operador Ternario



Sentencias condicionales

Ahora que sabemos como funciona la lógica de los condicionales, podríamos “achicar” un poco más esas funciones. Es decir, podemos escribir la misma instrucción pero de un modo más corto y prolijo.

El operador ternario, también conocido como operador condicional, es un operador que permite realizar una evaluación condicional de forma más concisa y expresiva en JavaScript.





Sentencias condicionales



El operador ternario consta de tres partes:

La condición es una expresión que se evalúa como verdadera (true) o falsa (false).

La expresión1 se ejecuta si la condición es verdadera.

La expresión2 se ejecuta si la condición es falsa.

```
condicion ? expresion1 : expresion2;
```





Sentencias condicionales

El operador ternario consta de tres partes:

Para utilizar los operadores ternarios, ya no vamos a escribir if...else, pero, su estructura se mantiene. Es decir, tenemos una condición ($\text{num1} > \text{num2}$), nuestro resultado por si esta condición es verdadero, lo asignamos utilizando el signo (?). Y para terminar, los dos puntos (:) nos permiten identificar cuál será su resultado falso.



```
let num1 = 7;
let num2 = 5;

//Operador ternario: (condición ?
verdadero : falso)

let rta = num1 > num2 ? "el 1" : "el 2";

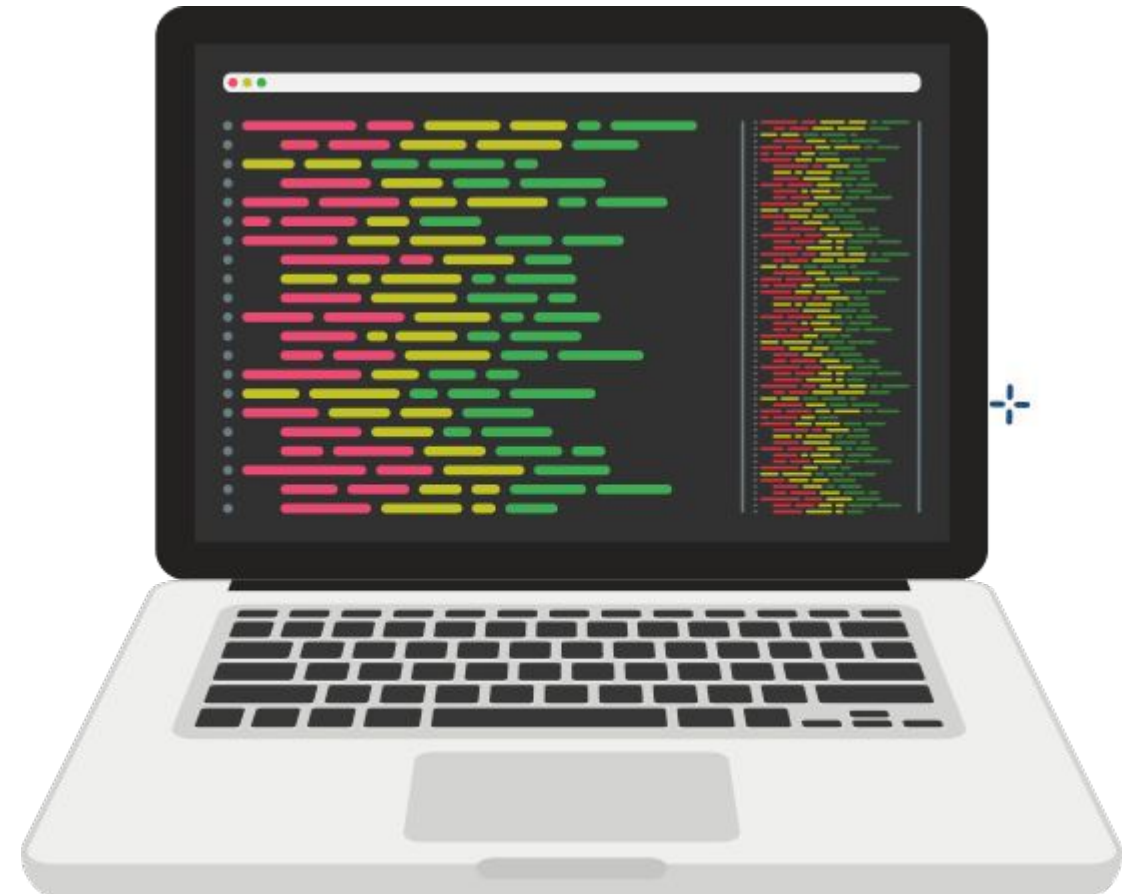
console.log(rta);
```

› Switch



Switch

La sentencia switch en JavaScript se utiliza cuando se necesita evaluar una expresión y ejecutar diferentes bloques de código según el valor de esa expresión. Es una forma más concisa y estructurada de manejar múltiples casos que cumplen ciertas condiciones.





Switch

expresion es la expresión que se va a evaluar.

case valor1: es un caso específico que se compara con la expresión. Si la expresión coincide con valor1, se ejecuta el código dentro de ese caso.

break; se utiliza para salir del switch después de ejecutar el código de un caso.

default: es un caso opcional que se ejecuta si ninguno de los casos coincide con la expresión.

```
switch (expresion) {  
    case valor1:  
        //Bloque de código a ejecutar si la expresion es igual a valor1  
        break;  
    case valor2:  
        //Bloque de código a ejecutar si la expresion es igual a valor2  
        break;  
    case valor3:  
        //Bloque de código a ejecutar si la expresion es igual a valor3  
        break;  
    // ...  
    default:  
        //Bloque de código a ejecutar si ninguno de los casos se cumple  
}
```



Switch

La expresión es evaluada una vez y luego se compara con cada uno de los casos especificados. Si hay una coincidencia entre la expresión y un caso, se ejecuta el bloque de código correspondiente.

Si ninguno de los casos coincide con el valor de la expresión, se ejecuta el bloque de código dentro del default.

El bloque default es opcional y se utiliza para manejar el caso en que no se cumpla ninguna de las condiciones anteriores.



```
let dia = "lunes";

switch (dia) {
  case "lunes":
    console.log("Hoy es lunes");
    break;
  case "martes":
    console.log("Hoy es martes");
    break;
  case "miércoles":
    console.log("Hoy es miércoles");
    break;
  default:
    console.log("Hoy no es ninguno de los días de la semana");
}
```

LIVE CODING

Ejemplo en vivo

Usando Switch

Eres un instructor de programación y estás enseñando cómo utilizar la estructura switch en JavaScript para tomar decisiones basadas en valores específicos.

Realizar una breve sesión de live coding para demostrar cómo utilizar la estructura switch en JavaScript para determinar el día de la semana según un número ingresado por el usuario.

 **Tiempo: 10 minutos**

LIVE CODING

Ejemplo en vivo

Paso 1: Obtener el Día de la Semana del Usuario

- Crea un nuevo archivo JavaScript (puedes nombrarlo como "diasemana.js").
- Dentro del archivo, declara una variable `numeroDia` para almacenar el número que el usuario ingresará para representar el día de la semana.
- Utiliza la función `prompt` para solicitar al usuario un número del 1 al 7 y asigna el valor ingresado a la variable `numeroDia`.

Paso 2: Utilizar la Estructura `switch`

- A continuación, utiliza la estructura `switch` para determinar el día de la semana en función del valor de `numeroDia`.
- Debes definir casos para los números del 1 al 7, y un caso por defecto para manejar valores no válidos.
- Mostrar el resultado al usuario utilizando `console.log()`.

Evaluación Integradora ✨

Trabajo Integrador del Módulo 💪

Iremos completándolo progresivamente clase a clase





Ejercicio N° 1

Estructura de control



Estructura de control

Breve descripción

Contexto: 🙌

Eres un desarrollador de software y te han asignado la tarea de crear un sistema de inicio de sesión simple utilizando JavaScript y la estructura de control if.

Consigna: 📝

Crear un sistema de inicio de sesión que solicita al usuario un nombre de usuario y una contraseña, verifica si son correctos y muestra un mensaje de bienvenida o error en función de la verificación.

Tiempo🕒: 15 Minutos





Estructura de control

Breve descripción

Paso a paso:

1. Crea un nuevo archivo JavaScript (puedes nombrarlo como "login.js").
2. Declara dos variables: `usuarioValido` y `contrasenaValida`. Asigna un nombre de usuario y una contraseña válidos como cadenas de texto.
3. Utiliza la función `prompt` para solicitar al usuario que ingrese su nombre de usuario y contraseña. Almacena estos valores en variables.
4. Utiliza la estructura de control `if` para verificar si el `usuarioIngresado` y la `contrasenaIngresada` coinciden con las credenciales válidas (`usuarioValido` y `contrasenaValida`).
5. Utiliza `console.log` para mostrar un mensaje de bienvenida si las credenciales son correctas o un mensaje de error si son incorrectas.





Estructura de control

Breve descripción

```
let usuarioValido = "usuario123";
let contrasenaValida = "claveSecreta";

let usuarioIngresado = prompt("Ingresa tu nombre de usuario:");
let contrasenaIngresada = prompt("Ingresa tu contraseña:");
if (usuarioIngresado === usuarioValido && contrasenaIngresada === contrasenaValida) {
  console.log(";Bienvenido, " + usuarioValido + "!");
} else {
  console.log("Credenciales incorrectas. Por favor, inténtalo nuevamente.");
}
```



○

¿Alguna consulta?

+



RESUMEN

¿Qué logramos en esta clase?

- ✓ Comprender cómo funcionan las expresiones aritméticas en JavaScript y cómo realizar operaciones matemáticas básicas como suma, resta, multiplicación y división.
- + ✓ Saber cómo utilizar operadores de asignación para actualizar el valor de una variable en función de una operación aritmética, como += para suma. +
- ✓ Puder utilizar la sentencia if para ejecutar un bloque de código si una condición especificada es verdadera.
- ✓ Comprender cómo funciona la sentencia if-else para ejecutar diferentes bloques de código en función de si una condición es verdadera o falsa.
- ✓ Estár familiarizado con la estructura switch y cómo se utiliza para realizar múltiples comprobaciones de igualdad en función de un valor.



#WorkingTime

Continuemos ejercitando

¡Antes de cerrar la clase! Te invitamos a: 🙌 🙌 🙌

1. Repasar nuevamente la grabación de esta clase
2. Revisar el material compartido en la plataforma de Moodle (lo que se vio en clase y algún ejercicio adicional)
 - a. Material 1: Lección 5: Bases del lenguaje Javascript: 25-38
3. Traer al próximo encuentro, todas tus dudas y consultas para verlas antes de iniciar nuevo tema.

¡Muchas Gracias!

Nos vemos en la próxima clase 🙌



Momento: ✚

Time-out!

🕒 5 min.

