

{desafío}
latam_

Introducción al lenguaje JavaScript _

Parte II



Control de flujos

- Codificar rutinas utilizando operadores y control de flujo.
- Codificar un programa en JavaScript utilizando estilos y convenciones de programación tales como indentación y estructura de código.
- Transformar flujos de if/else a switch y al revés.
- Refactorizar flujos de múltiples if y else utilizando operadores lógicos.

Competencias

Operadores y Comparadores

Operador	Símbolo
Asignación	=
Matemática	+ (adición), - (sustracción), * (multiplicación), / (división), % (módulo)
Asignación compuesta	+=, -=, *=, /=
Incrementar/Decrementar	++, --
Igualdad	== (sueltamente iguales) === (estrictamente iguales) != (sueltamente desiguales) !== (estrictamente desiguales).

Operadores y Comparadores

Operador	Símbolo
Comparación	< (menor que) > (mayor que) <= (menor o igual a que) >= (mayor o igual a que)
Lógico	&& (y) (ó) ! (negación)

Realicemos ahora el siguiente ejemplo, partiendo que x es igual a 20 y z es igual a 13, ¿cuál será el resultado para la siguiente operación $(x-z) > (z-x)$, utilizando operadores y comparadores?

**Resolvamos
en conjunto**

Considerando las siguientes variables: $a = 10$ y $b = 5$,
¿cuál es el resultado de las siguientes operaciones
lógicas?

- $a > 5 \parallel b < 10$
- $a == 10 \&\& b == 5$
- $(a > 5 \parallel a < 15) \&\& b > 0$
- $!(a > 5) \&\& b < 10$
- $(a - 5) >= (a*b) - (b/a)$

**Ahora te toca a
ti**

La estructura If

- Esta estructura se emplea para tomar decisiones en función de una condición:
 - Si la condición se cumple, se ejecutan todas las instrucciones que estén dentro de {...}.
 - Si la condición no se cumple, no se ejecuta la instrucción contenida dentro de las llaves {...}

```
if(condicion) {  
    ...  
}
```


Solicitar al usuario que ingrese dos números enteros, comparar si el primer número es menor o igual que el segundo, para luego comprobar si el segundo número es mayor o igual que cero y finalmente comprobar si el primer número es menor que cero o distinto de cero.

**Resolvamos
en conjunto**

Realiza un programa en JavaScript implementado la estructura de control if, que solicite al usuario ingresar dos números, luego que compruebe si el número 1 es positivo, si el número dos es negativo y finalmente si el número 2 es mayor o igual que el número 1, indicando si se cumple cada condición mediante un alert.

A vertical decorative line on the right side of the slide, featuring a series of white and gray symbols: a closing curly brace '}', an '@' symbol, another closing curly brace '}', and a closing parenthesis ')'.

**Ahora te toca a
ti**

La importancia de las buenas prácticas

Dentro de las buenas prácticas se encuentran:

- Indentación: Es igual que la sangría, una serie de espacios en blanco al inicio del código. En JavaScript la indentación no es una obligación para que el código funcione correctamente.
- Agregar un espacio en blanco antes y después de cada operador (`a = 1`, y no `a=1`) y escribir una única instrucción por línea.

Resolvamos en conjunto

Modificar el siguiente código aplicando buenas prácticas, el ejemplo consiste en:

- Solicitar dos número enteros al usuario y comparar si el número 1 es mayor que el número 2, si el número 2 es positivo y finalmente si el número 1 es negativo.

```
var num1=prompt("Ingresa el primer número: "),num2=prompt("Ingresa el segundo número: ");num1<num2&&console.log("El número 1 no es mayor que numero2"),num2>=0&&console.log("El número 2 es positivo"),num1<0&&console.log("El número 1 es negativo");
```

Estructura If - else

Si la condición que tenemos declarada se cumple, se ejecutan todas las instrucciones que se encuentran dentro de las llaves del if, pero si la condición es el caso contrario y no se cumple, se ejecutan todas las instrucciones contenidas en las llaves del else {}

```
if(condicion) {  
    ...  
}  
else {  
    ...  
}
```

Solicitar al usuario que responda a una pregunta en específico, en este caso: “¿Quiere usted aprender a programar con JavaScript?”, si el usuario responde que “Si” o “SI” o “si”, enviar el mensaje: “Felicitaciones, ya eres parte de Desafío Latam”, de lo contrario: “Que lastima!!!... te esperamos”.

A vertical decorative line on the right side of the slide, featuring a series of white and gray symbols: a closing curly brace '}', an '@' symbol, an opening curly brace '{', a sad face ':(', and a stylized figure of a person sitting at a desk.

**Resolvamos
en conjunto**

Realiza un programa en JavaScript implementado la estructura de control if-else, que solicite al usuario indicar si desea aprender a programar con NodeJS, si el usuario indica que “Si” o “SI” o “si” indicar mediante un alert “Es hora de estudiar NodeJS”, de lo contrario indicar mediante un alert “No quieres aprender NodeJS, sigue estudiando JS”.

A vertical line separates the white text area from the orange background. It features a series of white and grey symbols: a closing curly brace '}', an '@' symbol, another closing curly brace '}', and a closing parenthesis ')'.

**Ahora te toca a
ti**

Estructura Switch

Está especialmente diseñada para manejar de forma sencilla múltiples condiciones sobre la misma variable.

```
switch (variable) {  
    case valor_1:  
        // instrucciones  
        break;  
    case valor_2:  
        // instrucciones  
        break;  
    case valor_3:  
        // instrucciones  
        break;  
    default:  
        // instrucciones  
        break;  
}
```


Se solicita a un usuario que ingrese un número del 1 al 7
e indicar mediante un mensaje el día de la semana a cual
pertenece ese número

**Resolvamos
en conjunto**

Realiza un programa en JavaScript implementado la estructura de control switch, donde se solicita a un usuario que ingrese un número del 1 al 12 e indicar mediante un mensaje el mes del año al cual pertenece ese número. Si no pertenece a ningún mes indicar que el número no está permitido.

A vertical decorative line on the right side of the slide, featuring a series of white and gray symbols: a closing curly brace '}', an '@' symbol, another closing curly brace '}', and a stylized '@' symbol with a dot inside, resembling a person's head. The line is set against a solid orange background.

**Ahora te toca a
ti**

Resolvamos en conjunto

Se solicita al usuario que ingrese un número del 1 al 4, si el número es 1, se debe indicar al usuario que ingreso el número 1 y así sucesivamente, pero en el caso que el número no esté incluido entre el 1 y el 4, también se debe notificar

A vertical decorative line on the right side of the slide, featuring a series of overlapping symbols: a closing curly brace '}', an '@' symbol, another closing curly brace '}', and a closing parenthesis ')'.


**Transformar
múltiples
ciclos if-else
en switch**

Solicitar al usuario un número del 1 al 4 con las características de: 1-Suma, 2-Resta, 3-Multiplicacion, 4-Division, para así generar una respuesta dependiendo del número ingresado, si el número es uno, indicar mediante un alert que seleccionó sumar, si el número es 2 indicar mediante un alert que seleccionó restar y así sucesivamente. Por lo tanto, crea una estructura switch. que reciba el número de la opción y en base a eso, indique la opción seleccionada.

**Ahora te toca a
ti**

Resolvamos en conjunto

Se le solicita al usuario ingresar la edad y responder a la pregunta: “¿Quieres estudiar en Desafío Latam?”, para luego evaluar si la edad es mayor o igual a 18 y la respuesta es “si” o “Si” o “SI”, responder con un mensaje de bienvenida, de lo contrario, si la edad es menor que 18 y si quiere estudiar, responder con el mensaje: “Que bueno, pero debes estar acompañado por tu representante”, mientras que al indicar que “no” o “No” o “NO” desea estudiar en Desafío Latam, responder con el mensaje: “Que lastima!!!... te esperamos pronto”, y en el caso de no ser ninguno de los datos mencionados anteriormente, indicar un mensaje: “Datos errados”.



**Uso de
sentencias
condicionales
dentro de un
bloque if**

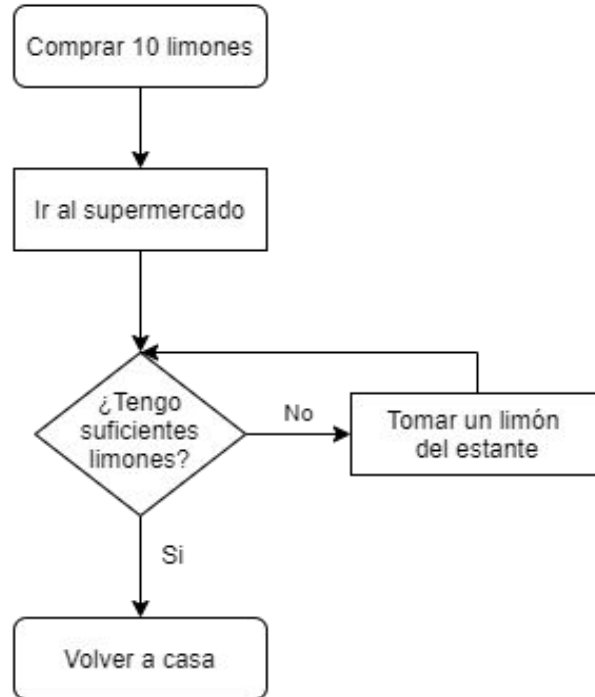
Ahora te toca a ti

Desarrolla un programa en JavaScript donde se solicite al usuario ingresar su edad y responder a la pregunta: “¿Estás estudiando en la universidad u otra institución educativa?” con un “Si” o “SI” o “si” o “NO” o “no” o “No”. Para luego evaluar si la edad es mayor o igual a 18 y la respuesta es “si” o “Si” o “SI”, responder con el mensaje: “Es correcto, deberías estar estudiando”, de lo contrario, si la edad es menor que 18 y si está estudiando, responder con el mensaje: “Que bueno, debes estar en el colegio aun, sigue así”, mientras que al indicar que “no” o “No” o “NO” está estudiando, responder con el mensaje: “Que mal!!!... deberías estar estudiando”, y en el caso de no ser ninguno de los datos mencionado anteriormente, indicar un mensaje: “Datos errados”

Ciclos

- Conocer los ciclos y su aplicación en la programación con JavaScript.
- Crear diagramas de flujo con iteraciones e implementarlos en JavaScript.
- Determinar cuándo se deben ocupar ciclos y cuando ifs.
- Diferenciar iteradores, contadores y acumuladores.
- Codificar rutinas utilizando estructuras de control repetitiva.

Diagrama de flujo con ciclos



Ciclo For

```
for (inicializacion; condicion; actualizacion) {  
    ...  
}
```

El Ciclo For nos permite realizar acciones que se repitan mientras se mantiene una condición.

1. Mostrar 10 veces en el documento web el mensaje: "{desafío} latam_".
2. Desarrollar y mostrar la tabla de multiplicar para el número 6

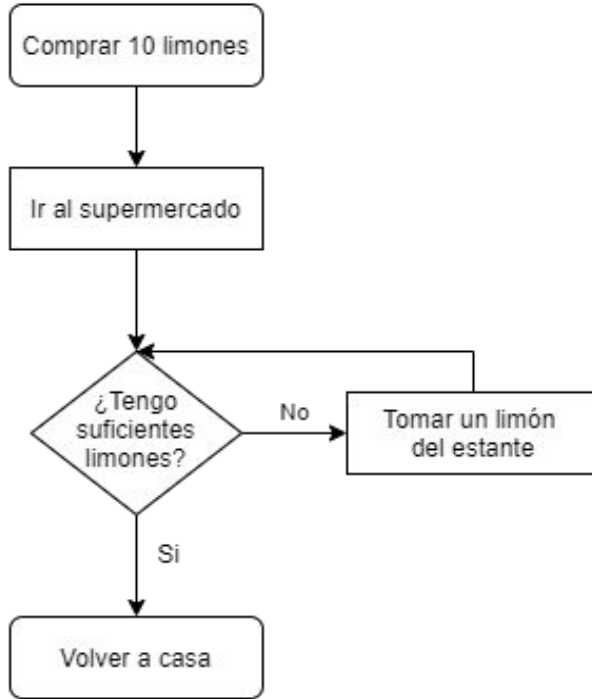
{desafío}
latam_

**Resolvamos
en conjunto**

Realizar un programa en JavaScript mediante el uso del ciclo for que permita recibir un número entero cualquiera y muestre la tabla de multiplicar del 1 al 10 para ese número ingresado. Además, el mensaje a mostrar debe tener la estructura: 1 X 10 = 10, para cada uno de los resultados mostrados, haciendo uso de interpolación en vez de concatenación.

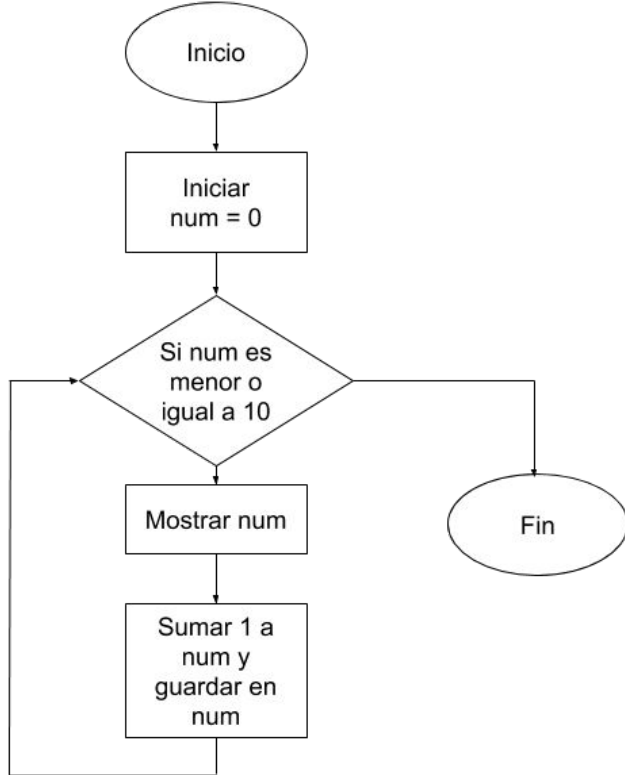
**Ahora te toca a
ti**

Resolvamos en conjunto



De diagrama
de flujo a
código
JavaScript

Implementa el siguiente diagrama de flujo usando el ciclo for.



Ahora te toca a ti

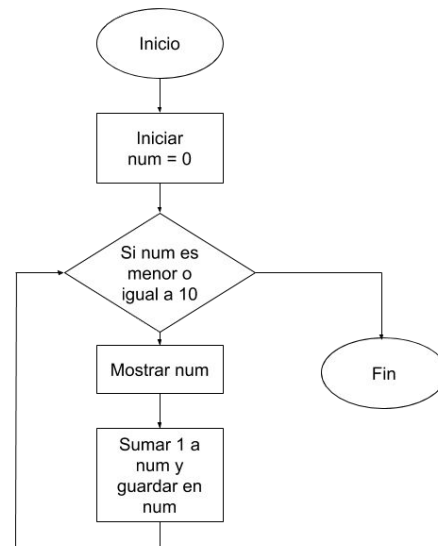
Ciclo while

```
while (condición) {  
    ...  
}
```

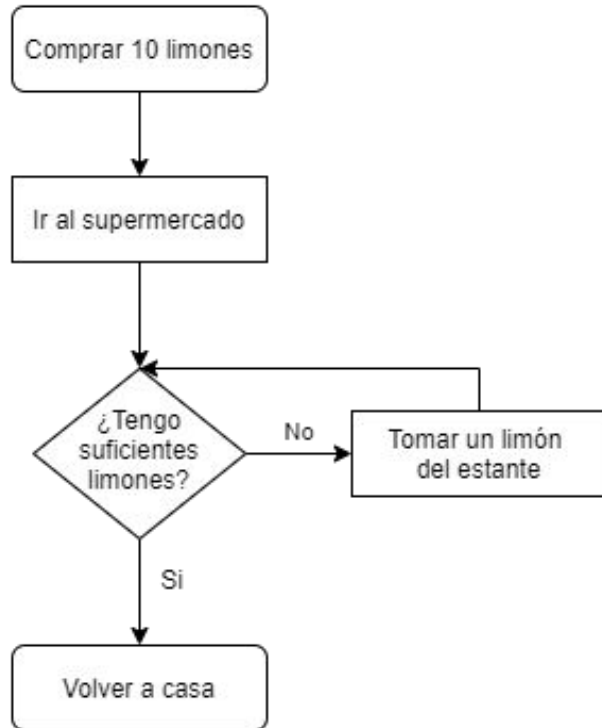
Mientras se cumpla una condición y sea verdadera se ejecutan las instrucciones que estén dentro del ciclo.

Resolvamos en conjunto

En el siguiente diagrama de flujo, se muestra como resultado en pantalla el listado de números desde el 0 hasta el 10, con una variable de inicio igual a 0 que servirá para contar y verificar la condición en el ciclo. Llevar el diagrama de flujo a un código JavaScript.



Implementar el diagrama de flujo del inicio del capítulo pero esta vez usando el ciclo while.



Ahora te toca a
ti

Ciclo do-while

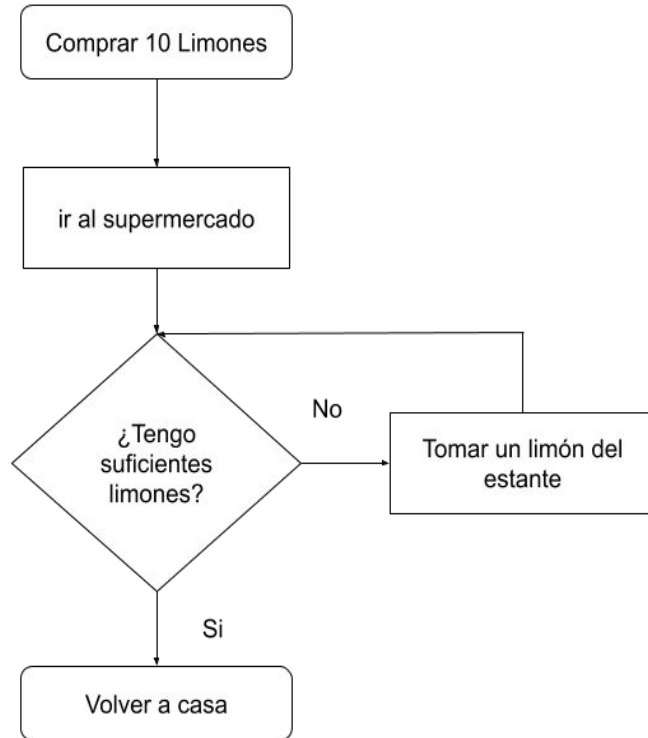
```
do {  
  
}  
while (condición);
```

Ejecuta la instrucción, luego ve la condición y si es falsa sale inmediatamente del bucle. Siempre va a ejecutar una instrucción al menos una vez.

Imprimir los número impares que correspondan desde el
cero (0) hasta el 20, utilizando do-while.

**Resolvamos
en conjunto**

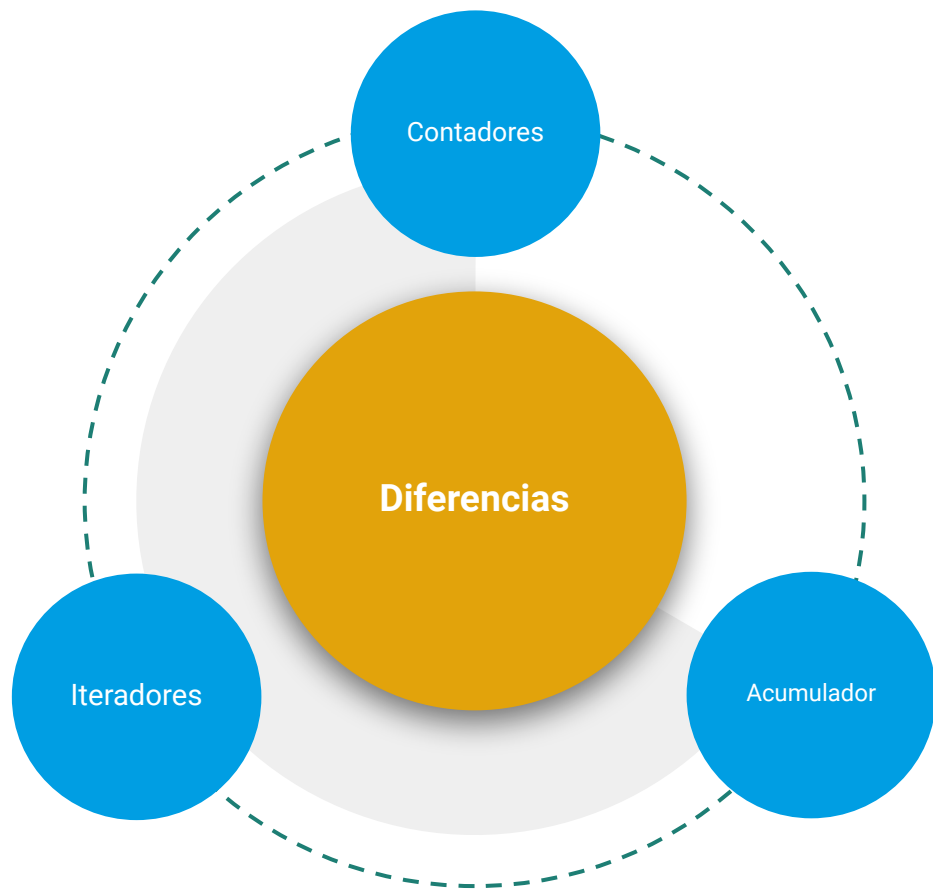
Implementa el diagrama de flujo del inicio del capítulo pero esta vez usando el ciclo do-while.



Ahora te toca a
ti

Diferencias entre un ciclo y un if

- Si se requiere verificar una única vez una condición, debes usar if.
- Si se requiere verificar múltiples veces una condición, entonces se debe usar ciclos repetitivos, como for o while.



Elaborar un programa con JavaScript que permita sumar e indicar la cantidad de números impares comprendidos entre el 0 y el 50 (incluidos ambos) implementando el ciclo for. Es decir, se deben sumar solamente los números impares entre el 1 y el 50, indica cuánto fue la suma y a su vez indicar cuántos números se sumaron.

**Resolvamos
en conjunto**

Elaborar un programa con JavaScript que permita sumar e indicar la cantidad de números pares comprendidos entre el 0 y el 100 (incluidos ambos) implementando el ciclo for. Es decir, se deben sumar solamente los números pares entre el 0 y el 100, indica cuándo fue la suma y a su vez indicar cuántos números se sumaron. (Ayuda: debes implementar el ciclo for, contadores y acumuladores).

**Ahora te toca a
ti**

Funciones en JavaScript

- Entender que son y para que se usan las funciones.
- Ser capaz de dividir problemas en subprocessos.
- Aplicar funciones para resolver problemas de carácter repetitivo o que sea necesario segmentar.
- Identificar entradas y salidas de una función.

Competencias

Funciones

- Son fragmentos de código que podemos reutilizar cuantas veces queramos dándoles dinamismo para que estas ejecuten o hagan cosas por nosotros.
- Son mini programas que cumplen una “función” determinada y pueden ser reutilizadas.

```
function nombre (parametros) {  
    // instrucciones  
}
```

Solicitar al usuario que ingrese la edad y dentro de una función se muestre la edad del usuario con un saludo desde el documento web.

A vertical line separates the white left side from the orange right side. It features a series of white and grey symbols: a closing curly brace '}', an '@' symbol, another closing curly brace '}', and a stylized '@' symbol that resembles a person's head and shoulders.


**Resolvamos
en conjunto**

Desarrollar un programa en javaScript que, solicitando al usuario el nombre, apellido y edad, mostrando así un saludo en el documento con el nombre, apellido y la edad, ejemplo: “Hola Juan Lopez, tienes 34 años”.

**Ahora te toca a
ti...**

Resolvamos en conjunto

Se debe solicitar al usuario dos números enteros y almacenarlos en variables distintas (valores de inicio), luego mediante el uso de una función, enviar los dos números ingresados a la función (argumento, llamado de la función y valores de entrada), para ser recibidos (parámetros) y procesados (sumando ambos números, procesamiento de valores) y finalmente retornar el valor final de la suma de ambos números (retorno de información y valores de salida)



**Dividir
problemas en
subprocesos e
identificar la
entrada y
salida de la
función.**


Desarrollar un programa en javaScript que, solicitando al usuario tres números enteros, realice la suma de los tres números y retorne el valor desde la función. Por lo tanto, se debe imprimir en pantalla el resultado de la función y enviar como argumentos los tres números ingresados por el usuario.

A vertical line separates the white left side from the orange right side. Along this line, there are several large, stylized symbols: a closing curly brace '}', an '@' symbol, another closing curly brace '}', and a closing parenthesis ')'.

**Ahora te toca a
ti**

Resolvamos en conjunto

Ahora haremos el mismo ejemplo anterior, pero lo utilizaremos en un mini formulario para poder segmentar y dejar la página siempre disponible para que el usuario interactúe con ella sin la necesidad de recargar o refrescar la página, es decir, el usuario deberá ahora ingresar los números en las entradas indicadas del formulario

A vertical line separates the white left side from the orange right side. It features a series of white symbols: a closing curly brace '}', an '@' symbol, an opening curly brace '{', and a stylized person icon with arms raised.

**Aplicar
funciones para
resolver
problemas de
carácter
repetitivo o que
sea necesario
segmentar.**

Desarrollar un programa en JavaScript que, solicitando un solo número al usuario, verifique mediante una función, si dicho número que ingresó el usuario es positivo, negativo o nulo. Por lo tanto, se debe indicar en caso positivo: “El número es positivo”. En el caso de ser negativo: “El número es negativo”. Si resulta ser nulo: “El número es nulo”.. (Ayuda: utiliza `parseInt` para convertir de string a number). Se debe enviar como argumento a la función el número ingresado por el usuario y el retorno de la función será la respuesta dependiendo de la evaluación del valor.

A vertical line separates the white text area from the orange background. It features a series of white and grey symbols: a closing curly brace '}', an '@' symbol, an opening curly brace '{', and a closing parenthesis ')'.

**Ahora te toca a
ti**

{desafío}
latam_

*Academia de
talentos digitales*

www.desafiolatam.com