

Lógica de programación:

Sumérgete en la programación con JavaScript

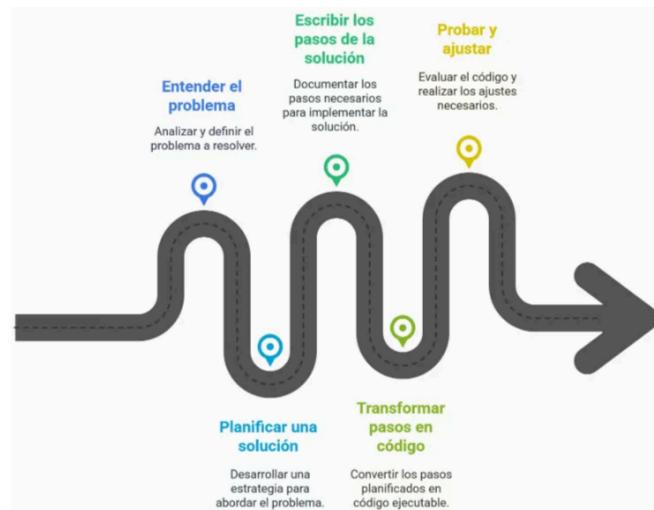
Índice:

1. Iniciando con JavaScript
 2. Condicionales y concatenación
 3. Loops y tentativas
 4. Buenas prácticas en programación
 5. Desafío
-

Iniciando con JavaScript

1. ¿Qué es la lógica de la programación?

Es la capacidad de organizar y estructurar ideas para resolver problemas a través del código



Aprendemos la lógica en programación porque es la base del razonamiento computacional. Te enseña a pensar de manera clara, objetiva y estructurada.

Como por ejemplo, los pasos que uno tiene para abrochar un zapato: tomar los calcetines, tomar el par de zapatos, sentarse, ponerte un calcetín a la vez, poner los zapatos, ajustar y atar.

2. ¿Qué es un lenguaje de programación?

Las computadoras operan con impulsos eléctricos que se representan mediante código binario (0 y 1). Para comunicarnos con las máquinas utilizamos el lenguaje de programación.

Este lenguaje es un conjunto de palabras, símbolos y reglas que permiten escribir instrucciones para que la computadora pueda interpretar y ejecutar.

Existe hasta un lenguaje de Emojis <https://github.com/emojicode/emojicode>

Errar es parte del proceso de aprender a programar. Debemos ser pro activos en investigar, buscar, leer, etc. Tomaremos un rol de detectives que buscan la solución ante el problema.

JavaScript es:

- El lenguaje más popular y utilizado
- Funciona directamente en el navegador
- Permite ver el resultado del código casi en tiempo real
- Es una excelente puerta de entrada para el desarrollo web y para aprender lógica.

3. Proyecto inicial

Herramientas: Visual Studio Code

4. Preparando el ambiente

Instalar VSCode.

A diferencia de otros lenguajes que necesitan programas específicos para ejecutarse, JavaScript funciona directamente en el navegador. Es decir, además de VS Code, no necesitas instalar nada para comenzar a probar tu código. Solo necesitas tener un navegador instalado — como Google Chrome, Firefox, Edge u Opera — y ya podrás ejecutar tus primeros scripts directamente en la pantalla.

El proyecto base contiene:

- Una carpeta con **imágenes** que se usarán en actividades futuras;
- Un archivo **HTML**, responsable de estructurar el sitio y conectar los otros archivos;
- Un archivo **CSS**, que define el estilo visual de la página (colores, fuentes, tamaños, etc.);
- Un archivo **JavaScript**, donde vamos a escribir nuestro código y ejercicios.

Aunque el proyecto trae varios archivos, en este momento nos enfocaremos solo en el archivo JavaScript, ya que es en él donde pondrás en práctica todo lo que estás aprendiendo sobre lógica de programación.

Después de descargar el proyecto abriremos VSCode:

- Ve al menú **Archivo > Abrir Carpeta** (o `File > Open Folder`, si el VS Code está en inglés).
- Pega la ruta de la carpeta copiada anteriormente.
- Haz clic en **Seleccionar Carpeta**.

¿Cómo abrir el proyecto en el navegador?

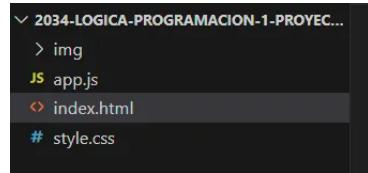
Lo que realizamos en el VSCode con código en JavaScript, lo hacemos del archivo index.html en el navegador. Esto sucede porque el navegador necesita un **punto de entrada** para cargar toda la página, y ese punto es precisamente el HTML. El archivo HTML es el responsable de estructurar el contenido de la página e indicar al navegador qué archivos adicionales debe cargar, como:

- El **CSS**, que define el estilo (colores, fuentes, tamaños, etc.);
- El **JavaScript**, que define el comportamiento y la lógica de la aplicación.

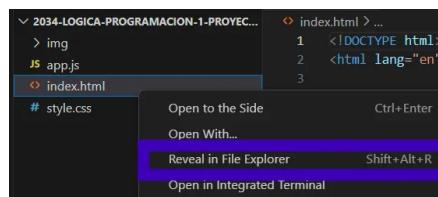
Es decir, **es el HTML el que llama al JavaScript**, y por eso, es él que necesitamos abrir para ver el resultado de nuestro código funcionando en la práctica. Así que:

1. En **Visual Studio Code**, localiza el archivo `index.html` dentro de la carpeta del proyecto.

2. Haz clic derecho sobre él.

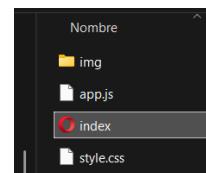


3. Selecciona la opción "**Revelar en el Explorador de Archivos**" (o "**Reveal in File Explorer**", si el VS Code está en inglés).

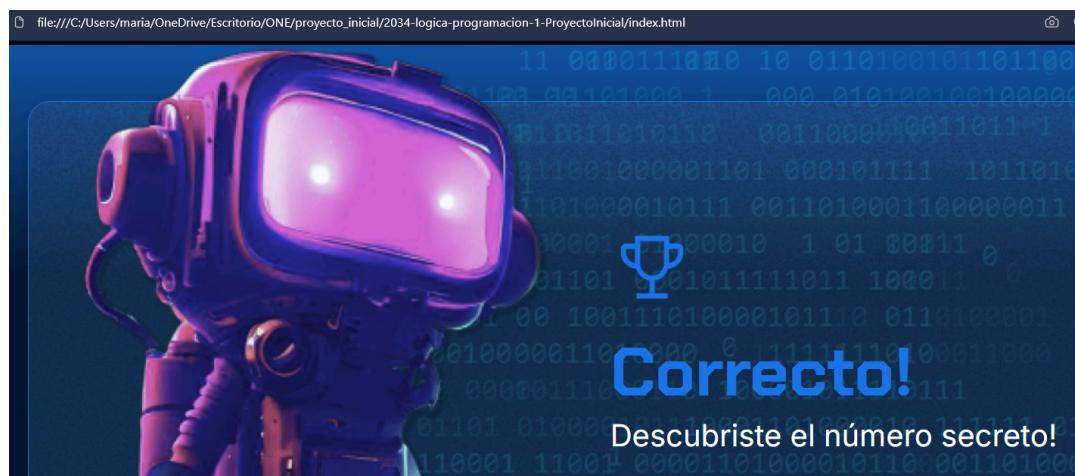


4. La carpeta se abrirá en tu computadora.

Ahora, **haz doble clic en el archivo `index.html`**.



5. ¡Listo! El navegador predeterminado de tu sistema se abrirá con el proyecto cargado.



Siempre guardar el archivo ante cualquier cambio (Ctrl + S)

Actualizar la pagina en el navegador (F5 o en recargar)

5. Variable

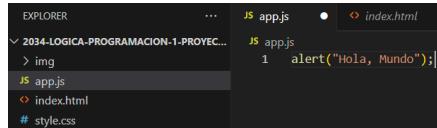
a. Tenemos 3 archivos en el proyecto:

- i. `index.html`: Renderiza o construye la interface
- ii. `style.css`: Estilos, formas de la interfaz
- iii. `app.js`: Es donde trabajaremos en este proyecto

El primer contacto es el tradicional "Hola Mundo" y se hace de la siguiente manera:

En app.js debe poner:

```
alert('Hola Mundo');
```



Las " es lo que queremos que se vea en el navegador (comillas simple o comillas dobles).

el ; es necesario poner en cada uno de las sentencias

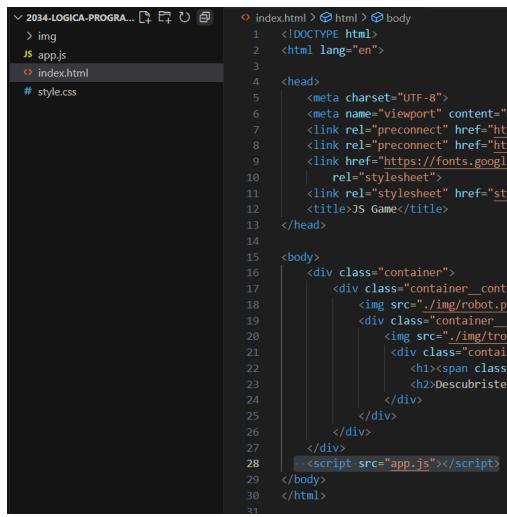
alert: es un mensaje de alerta o un pop -up (ventana emergente)

Para integrar lo nuevo que estamos trabajando, es necesario vincular o agregar lo que tenemos en la pestaña de app.js al index.html. En este caso del proyecto, lo agregaremos al final del código de index.html. Pero existen diferentes casos donde es recomendable ubicarlos en ciertas líneas de código específicas como en la cabecera.

```
<script src="app.js"></script>
```

script:

src: atributo, origen (source), donde tener el archivo



Para continuar con el proyecto debemos recordar que tenemos que automatizar el juego (adivinador de número) y para esto, necesitamos interacción con el usuario. Para esto tenemos el método prompt, la cual nos permite hacer preguntas al usuario, podemos enviar un mensaje para que el usuario complete la información solicitada.

Cuando el usuario entrega el valor ¿Dónde va? A una variable (espacio en memoria, como un papel, caja que recibe datos) Para esto, debemos indicársela a JS y se hace con 3 palabras claves que van a definir el como se comportara esa caja que recibe datos. Estas son: const, let y var. En este caso vamos a utilizar "let" (mas adelante se estudiaran estas variables).

Para poder llegar a esta caja, necesitamos saber como se llama, para esto a las variables hay que asignarles un nombres y una forma correcta de llamarlas es:

- camel case (notación de camello): siempre escribir las en minúscula y si la variable tiene más de una palabra, la primera letra de la segunda palabra y las palabras sucesivas siempre se coloca en mayúsculas. (let numeroUsuario) (numeroDeUsuario)

- RECORDAR: Utilizar nomenclaturas representativas y auto explicativas

The screenshot shows a file tree on the left with files: 2034-LOGICA-PROGRAMACION.html, img, app.js, index.html, and style.css. The app.js file is open on the right, containing the following code:

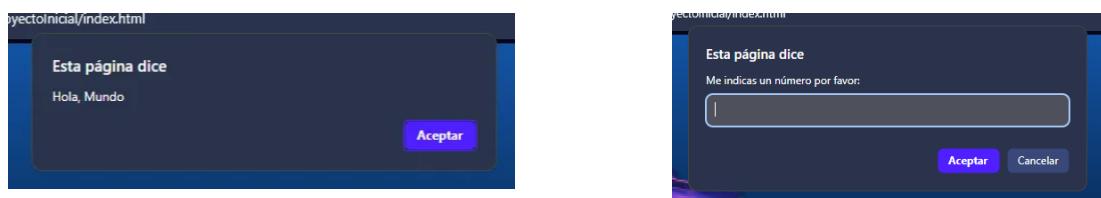
```
JS app.js > ...
1 alert("Hola, Mundo");
2 let numeroUsuario = prompt("Me indicas un número por favor:");
```

Explicación de la sintaxis:

Cuando trabajamos con asignación o declaración de variables, la variable va al lado izquierdo. Vamos a encontrar el = (asignación), después al lado derecho encontraremos el valor que se le va atribuir.

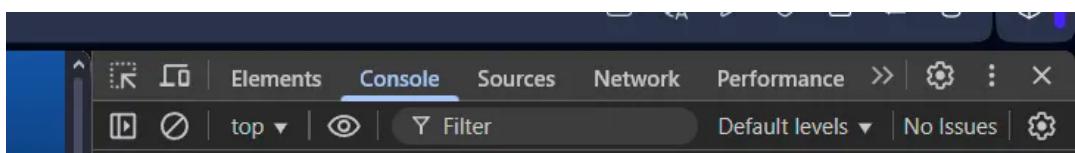
`prompt`: instrucción o pregunta, una ventana emergente (popup) al usuario, permite que el usuario escriba algún dato, devuelve el dato como un string

Variable creada: `numeroUsuario`



6. Condición if

Al construir nuestro `prompt` (Me indicas un número por favor) y definir nuestra variable (`numeroUsuario`), donde el usuario coloca el número, pero no sabemos dónde quedó almacenado esa respuesta (número), para esto verificamos en la consola del navegador.



Para unir esta información, debemos seguir trabajando en `app.js` y agregar el siguiente código

```
console.log(numeroUsuario);
```

Esto va sin comillas porque estamos haciendo referencia a una variable, en nuestro caso, `numeroUsuario`. Colocamos las "" cuando es un valor constante, uno que no cambiara.

Al agregar este código, la información entregada por el usuario será almacenada en la variable `numeroUsuario` y se puede observar que sale el número en la consola del navegador.

Para continuar con el proyecto, vamos a crear otra caja, donde almacenaremos la respuesta correcta

```
let numeroSecreto = 6;
```

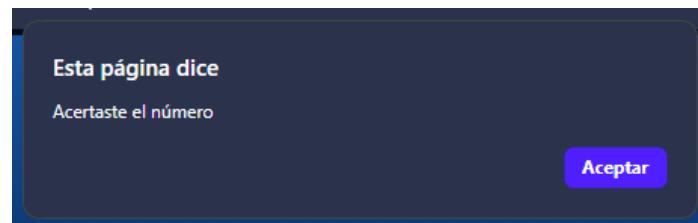
Esta respuesta será nombrada como variable `numeroSecreto`. Para comparar si el usuario apuestó la respuesta correcta (6) vamos a integrar la condición if

```
if (numeroUsuario == numeroSecreto){ alert ('Acertaste el número');}
```

== : Comparar

RECORDAR: Los let deben ir al inicio para mantener un orden

```
let numeroSecreto = 6;  
let numeroUsuario = prompt("Me indicas un número por favor:");  
console.log(numeroUsuario);  
if (numeroUsuario == numeroSecreto){ alert('Acertaste el número');}  
}
```



7. Alert y prompt

Una persona está desarrollando la pantalla de bienvenida de una aplicación hecha con JavaScript y ha creado el siguiente código:

```
let mensajeDeBienvenida = 'Bienvenido a la aplicación';
alert('mensajeDeBienvenida');
```

[COPIA EL CÓDIGO](#)

Con base en esto, analiza las afirmaciones a continuación y marca solo las verdaderas con respecto al código mostrado arriba.

A

Al ejecutar el programa, se mostrará un cuadro con el mensaje "Bienvenido al App".



El problema en este código es que la variable `mensajeDeBienvenida` no se está utilizando correctamente al mostrarla en la función `alert`.



En el código presentado, la línea `alert('mensajeDeBienvenida')` está pasando un texto y la variable no se está empleando de manera correcta.

C

Ningún mensaje se mostrará al ejecutar el programa.



Al ejecutar el programa, se mostrará un cuadro con el mensaje "mensajeDeBienvenida".



Para que se muestre el mensaje almacenado en la variable, es necesario eliminar las comillas simples, dejando el código del alert de la siguiente manera: `alert(mensajeDeBienvenida)`.

8. Cambiando el valor de las variables

En un proyecto práctico dentro del curso "Iniciando con JavaScript", estás desarrollando un juego en el cual los usuarios tienen que adivinar el número secreto. Has escrito un código que muestra un mensaje de bienvenida a los jugadores, les pide que elijan un número entre 1 y 10 y les da una respuesta si acierran, siguiendo la secuencia a continuación:

```
<br>javascript<br>alert('Bienvenidos al juego del número secreto');<br>let eleccion =  
prompt('Elige un número entre 1 y 10');<br><br>let numeroSecreto = 4;<br><br>if (eleccion ==  
numeroSecreto) {<br> alert('Adivinaste');<br>} <br>
```

¿Qué variable tendrías que cambiar para cambiar el número secreto en cada ronda del juego?

numeroSecreto



✓ ¡Lo lograste!

¡Hola, María! Tu respuesta es correcta, ¡felicitaciones! La variable que debes cambiar para modificar el número secreto en cada ronda del juego es `numeroSecreto`. Al cambiar el valor asignado a esta variable, el juego utilizará un nuevo número secreto que los jugadores deberán adivinar. ¡Sigue practicando y explorando las posibilidades de JavaScript!

9. ¿Dónde está el error?

Luis es un apasionado de la tecnología y está dando sus primeros pasos en el mundo de la programación. Para practicar la lógica de programación, decidió desarrollar un sistema de inicio de sesión de usuario único en el que solo se necesita proporcionar la contraseña para ver el contenido del resto del proyecto.

Sin embargo, al probar su proyecto, se dio cuenta de que no importaba la contraseña que ingresara en el sistema, siempre se le permitía el acceso.

Sabiendo que tú también estás estudiando lógica de programación, Luis te pidió ayuda para que puedan encontrar el error juntos. Para ello, ha proporcionado el código que escribió:

```
javascript
let contraseniaDelSistema = "contraseñaPrueba!";

let contrasenia = prompt("Ingrese la contraseña del sistema:");

if (contrasenia == contraseniaDelSistema){
    alert("Acceso al sistema garantizado");
}
```

[COPIA EL CÓDIGO](#)

De acuerdo con el código, ¿Qué cambio podrías sugerirle a Luis para que su sistema funcione como se imagina?

```
If (contrasenia == contraseniaDelSistema{
    alert("Acceso al sistema garantizado");
}
```



¡Lo lograste!

¡Hola, María! Tu respuesta es correcta, felicitaciones! El problema en el código de Luis es que está utilizando un operador de asignación (=) en lugar de un operador de comparación (==) dentro de la condición del 'if'. Al usar el operador de asignación, el valor de 'contraseñaDelSistema' se está asignando a 'contraseña', lo que siempre evalúa la condición como verdadera.
¡Sigue practicando y profundizando tus conocimientos en JavaScript, estás haciendo un gran trabajo!

10. Desafío: Hora de prácticas

Practicar la lógica de programación, incluyendo conceptos como variables, condicionales (if), alertas (alert), solicitudes (prompt), es esencial para tu carrera y desarrollo.

Estos fundamentos proporcionan la base para resolver problemas de manera estructurada, tomar decisiones en el código, crear bucles controlados e interactuar eficazmente con las personas.

Comprender estos conceptos no solo facilita el aprendizaje de nuevos lenguajes y tecnologías, sino que también te capacita para generar soluciones innovadoras, depurar de manera eficiente y mantener la calidad a lo largo del ciclo de vida del software.

Por lo tanto, invertir tiempo en estos principios desde el principio es fundamental para construir una carrera exitosa en el campo de la programación.

Con esto en mente, hemos creado una lista de actividades (no obligatorias) centradas en la práctica para mejorar aún más tu experiencia de aprendizaje. ¿Listo para practicar?

Desafíos

- a. Muestra una alerta con el mensaje "¡Bienvenida y bienvenido a nuestro sitio web!".

```
alert("¡Bienvenida y bienvenido a nuestro sitio web!");
```



- b. Declara una variable llamada nombre y asígnale el valor "Lua".

- c. Crea una variable llamada edad y asígnale el valor 25.

```
... JS app.js X index.html ...
JS app.js > ...
1 alert("¡Bienvenida y bienvenido a nuestro sitio web!");
2
3 let nombre = "Lua";
4 let edad = 25;
5
6 console.log (nombre, edad);
7
```

```
file:///C:/Users/maria/OneDrive/Escritorio/O ...
Elements Console Sources Network > Default levels No Issues
Lua 25 app.js:6
```

Nota: Al principio me arroja error porque no use las "", por ende JS interpreta Lua como variable y no texto.

- d. Establece una variable numeroDeVentas y asígnale el valor 50.

- e. Establece una variable saldoDisponible y asígnale el valor 1000.

```
JS app.js > ...
1 alert("¡Bienvenida y bienvenido a nuestro sitio web!");
2
3 let numeroDeVenta = 50;
4 let saldoDisponible = 1000;
5
6 console.log (numeroDeVenta, saldoDisponible);
7
```

```
//C:/Users/maria/OneDrive/Escritorio/ONE/prc ...
Elements Console Sources Network Performance > Default levels No Issues
50 1000 app.js:6
```

- f. Muestra una alerta con el texto "¡Error! Completa todos los campos".

```
JS app.js
1 alert("¡Error! Completa todos los campos");
2
3
```

g. Declara una variable llamada mensajeDeError y asígnale el valor "¡Error! Completa todos los campos". Ahora muestra una alerta con el valor de la variable mensajeDeError .



The screenshot shows a browser window with a dark theme. At the top, there is a tab labeled "Game" and a search bar. Below the address bar, the URL is displayed as "file:///C:/Users/maria/OneDrive/Escritorio/ONE/". A modal dialog box is open in the center of the screen. The dialog has a purple header bar with the text "Esta página dice" and a purple footer bar with the button "Aceptar". The main content area of the dialog contains the text "¡Error! Completa todos los campos" in white. The background of the browser shows a blurred image of a landscape.

```
JS app.js > ...
1 let mensajeDeError = "¡Error! Completa todos los campos";
2
3 alert(mensajeDeError);
4
```

h. Utiliza un prompt para preguntar el nombre del usuario y almacénalo en la variable nombre.

i. Pide al usuario que ingrese su edad usando un prompt y almacénala en la variable `edad`.

The screenshot shows a browser window with a modal dialog and two developer tool consoles.

Modal Dialog:

- Header: "Esta página dice"
- Text: "¿Cuál es tu nombre?"
- Input field: "Amaris"
- Buttons: "Aceptar" (Accept) and "Cancelar" (Cancel)

Developer Tools - Top Left:

- Console tab selected.
- Code editor pane with `app.js` containing:

```
1 let nombre = prompt("¿Cuál es tu nombre?");
2
3 console.log(nombre);
```
- Output pane below the code editor.

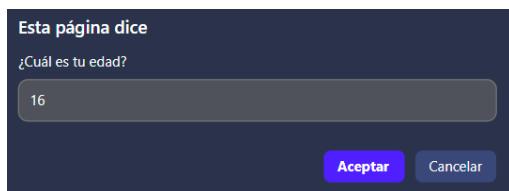
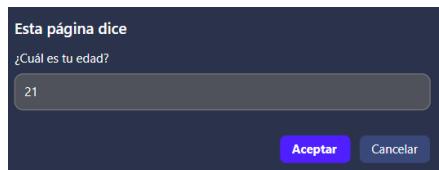
Developer Tools - Top Right:

- Console tab selected.
- Code editor pane with `app.js` containing:

```
1 let nombre = prompt("¿Cuál es tu nombre?");
2 let edad = prompt("¿Cuál es tu edad?");
3
4 console.log(nombre, edad);
```
- Output pane below the code editor showing the result: "Amaris 6".

j. Ahora, si la edad es mayor o igual a 18, muestra una alerta con el mensaje "¡Puedes obtener tu licencia de conducir!"

```
JS app.js > ...
1  let edad = prompt("¿Cuál es tu edad?");
2  edad = Number(edad);
3
4  if (edad>= 18){
5      alert("Puedes obtener tu licencia de conducir");
6  }
7  if (edad<17){
8      alert("No puedes obtener licencia de conducir");
9  }
10 console.log (edad);
```



Explicación:

```
let edad = prompt("¿Cuál es tu edad?");
```

Pide al usuario su edad y la guarda en `edad` como texto.

```
edad = Number(edad);
```

Convierte el valor de `edad` de texto a número para poder compararlo.

```
if (edad >= 18) { alert(...) }
```

Usa una estructura de control `if` para verificar si `edad` es mayor o igual a 18.

Si es verdadero, muestra la alerta con el mensaje.

11. Para saber más: documentación

a. <https://www.aluracursos.com/blog/guia-de-javascript>

i. ¿Qué es JS?

1. Lenguaje de programación
2. Destaca por dinamizar las páginas web

ii. ¿Dónde se ejecuta JS?

- i. El propósito principal es agregar interacción en las páginas
- ii. Se ejecuta en los navegadores

iii. Variables en JS

i. Una variable es un espacio en la memoria de la computadora que reserva el programa en ejecución. Usamos este espacio para almacenar información, realizar operaciones, etc.

ii. Escopo / scope: Lugar donde una variable existe y se puede usar.

- i. Escopo global: La variable existe en todo el archivo o script, se puede usar en cualquier parte del código.
- ii. Escopo local: La variable existe solo dentro de un bloque específico (una función `{ }` , un `if`, un `for`, etc). Fuera de ese bloque, no existe.

iii. Tiene 3 tipos de variables:

- i. `var`: Evita usar en proyectos modernos.
- ii. `let`: Se usa para variables que cambiarán de valor. Sus usos comunes son para contadores en "for", variables que cambian durante una función, estados intermedios.
- iii. `const`: Se usa para valores que no deben cambiar. Su uso más común es para constantes matemáticas, datos que no se deben modificar, referencias a objetos que no van a reasignarse

Mini resumen fácil de recordar			
Variable	¿Cambia valor?	¿Visible fuera del bloque?	¿Se usa hoy en día?
<code>var</code>	✓	✓ (si está en bloque)	⚠️ No recomendado
<code>let</code>	✓	✗	✓ Sí
<code>const</code>	✗	✗	✓ Sí

<https://www.aluracursos.com/blog/tipado-dinamico-con-javascript>

- b. https://developer.mozilla.org/es/docs/Learn_web_development/Core/Scripting/What_is_JavaScript

12. Lo que aprendimos

Aquí tienes un resumen de los temas enseñados en esta clase:

- Preparamos el entorno de desarrollo instalando Visual Studio Code para crear programas utilizando el lenguaje JavaScript.
- Comprendimos el concepto de variable para almacenar información, como números o palabras, para usarla más tarde en nuestro programa.
- Utilizamos el alert para mostrar un mensaje proporcionando algunas instrucciones sobre el programa, y utilizamos el prompt para interactuar con la persona, permitiéndole ingresar un valor y almacenándolo en una variable.
- Creamos un `if`, que es una instrucción en programación que permite que la computadora tome decisiones al ejecutar un bloque de código solo si una condición específica es verdadera.

En la próxima clase:

Seguiremos las buenas prácticas de programación y profundizaremos aún más en nuestros conocimientos.

Condicionales y concatenación

1. Proyecto de aula anterior

- a. Descargar una carpeta para trabajar

2. Comentarios y Else

- a. En JS podemos realizar comentarios en el código, ya sea para describir o dejar notas.
- b. Se utiliza: /* escribes el comentario */ o puede ser //escribe el comentario
- c. Else (*de otro modo*): Define una alternativa para ejecutar un bloque de código cuando la condición if no se cumple

```
JS app.js > ...
1 let numeroSecreto = 6;
2 let numeroUsuario = prompt("Me indicas un numero por favor:");
3
4 console.log(numeroUsuario);
5
6 if (numeroUsuario == numeroSecreto) {
7     alert('Acertaste el numero');
8 } else {
9     alert('Lo siento, no acertaste el número');
10 }
```

Como podemos observar, el comando ELSE nos esta dando la posibilidad de que el usuario este informado que su respuesta no es acorde a lo que el juego esta solicitando (6)

RECORDAR: Estamos programando un juego para que el usuario adivine el número secreto.

3. Template Strings

- Técnica que permite la interpolación de variables (insertar valores dentro del texto)
- Se utiliza `` y \${ }

```
JS app.js   X  index.html
JS app.js > ...
1 let numeroSecreto = 6;
2 let numeroUsuario = prompt("Me indicas un numero entre 1 y 10 por favor:");
3
4 console.log(numeroUsuario);
5
6 if (numeroUsuario == numeroSecreto) {
7     alert(`Acertaste, el número secreto es: ${numeroSecreto}`);
8 } else {
9     alert('Lo siento, no acertaste el número');
10 }
```

Como vemos en el código, utilizamos las el acento al revés para tomar toda la línea de texto que vera el usuario, para luego solo poner \${ } en la variable que queremos que el usuario pueda visualizar.

Esta página dice
Acertaste, el número secreto es: 6

Ejemplo mal ejecutado:

```
app.js > ...
1 let numeroSecreto = 6;
2 let numeroUsuario = prompt("Me indicas un numero entre 1 y 10 por favor:");
3
4 console.log(numeroUsuario);
5
6 if (numeroUsuario == numeroSecreto) {
7     alert('Acertaste, el número secreto es: numeroSecreto');
8 } else {
9     alert('Lo siento, no acertaste el número');
10 }
```

Esta página dice
Acertaste, el número secreto es: numeroSecreto

4. Live Server

- a. Visual code, tiene una herramienta que se llama extensión, la cual ocuparemos Live Server, nos ayuda a ver de manera en directo como funciona nuestro código .

5. Edad mínima para conducir

Eres un desarrollador de aplicaciones web y te han pedido evaluar un código que determina si un usuario cumple con la edad mínima requerida para conducir vehículos automotores. Según las normas establecidas, la edad mínima para conducir es de 18 años.

El siguiente código fue presentado como propuesta inicial:

```
let edad = prompt("Ingrese su edad");

if (edad >= 18) {
    console.log("Eres mayor de edad.");
} else {
    console.log("Eres menor de edad.");
}
```

[COPIA EL CÓDIGO](#)

Con base en este código, analiza las afirmaciones que se presentan a continuación y selecciona únicamente las correctas.

A

Si la edad es mayor que 18, el mensaje que se mostrará en la consola será: "Eres menor de edad".



C

Si la edad es menor que 18, el mensaje que se mostrará en la consola será: "Eres menor de edad".



Si la edad ingresada es menor que 18, la condición en el if será falsa (edad \geq 18 es falso), y el flujo de ejecución pasará al bloque del else, mostrando el mensaje "Eres menor de edad".

D

Si la edad es igual a 18, el mensaje que se mostrará en la consola será: "Eres mayor de edad".



Si la edad ingresada es igual a 18, la condición en el if será verdadera (edad \geq 18 es verdadero), y se mostrará el mensaje "Eres mayor de edad", ya que el operador \geq incluye el valor 18. Se ignorará el bloque de código dentro del else.

D

Si la edad es igual a 18, el mensaje que se mostrará en la consola será: "Eres menor de edad".



6. Cambiando el mensaje alert

Eres un desarrollador de software en una clínica médica llamada Médica Voll. La empresa desea que crees un pequeño juego de adivinanzas para sus pacientes mientras esperan en la sala de espera. Entonces, decides crear un juego del número secreto utilizando JavaScript, como se muestra a continuación:

```
javascript
alert('Bienvenido al juego del número secreto');
let intento = prompt('Elige un número entre 1 y 10');

let numeroSecreto = 4;

console.log(intento == numeroSecreto);
if (intento == numeroSecreto) {
    alert('Acertaste');
} else {
    alert('El número secreto era ' + numeroSecreto);
}
```

[COPIA EL CÓDIGO](#)

Dada esta estructura básica de código que creaste, ¿cómo modificarías el mensaje del alert para incluir el número que el paciente eligió en caso de no acertar el número secreto? Elige la opción correcta:

- A `alert('El número secreto era ' + numeroSecreto, ' y elegiste ' + intento)` 
- B `alert('El número secreto era ' + numeroSecreto + intento)` 
- C  `alert('El número secreto era ' + numeroSecreto + ', pero elegiste ' + intento);` 
Este código concatena correctamente el número secreto y la elección del paciente en el mensaje de alerta.
- D `alert('El número secreto era numeroSecreto y elegiste intento');` 

7. Trabajo con condicionales

Eres una persona recién llegada al equipo de desarrollo de *Jornada Millas*, un sitio web de compra de paquetes de viaje para los principales destinos del mundo.

Como primera tarea, tu líder te ha pedido que al introducir un número por teclado, muestre un mensaje en la consola que indique si el número es positivo, negativo o cero.

Selecciona la opción correcta:

A

```
const numero = prompt("Introduce un número:");

if (numero != 0) {
    console.log("El número es positivo o negativo");
} else {
    console.log("El número es cero");
}
```



```
const numero = prompt("Introduce un número:");

if (numero > 0) {
    console.log("El número es positivo");
} else if (numero < 0) {
    console.log("El número es negativo");
} else {
    console.log("El número es cero");
}
```

Si numero es mayor que 0, se ejecuta la primera instrucción y se muestra el mensaje "El número es positivo". Si numero es menor que 0, se ejecuta la segunda instrucción y se muestra el mensaje "El número es negativo". Si numero es igual a 0, se ejecuta la tercera instrucción y se muestra el mensaje "El número es cero".

C

```
const numero = prompt("Introduce un número:");

if (numero > 0) {
    console.log("El número es positivo");
} else {
    console.log("El número es negativo o cero");
}
```

8. Haga lo que hicimos en aula: `console.log`

El `console.log` es una función muy importante en lenguajes de programación, especialmente cuando se trabaja con JavaScript. Su función principal es imprimir mensajes en la consola del entorno de desarrollo, lo que permite probar información relevante durante la ejecución de un programa.

Ahora, incluya comandos `console.log` en diferentes partes del código para verificar el flujo del programa, los valores de las variables y otra información relevante durante la fase de desarrollo.

9. Desafío: hora de practicar

- Pregunte al usuario qué día de la semana es. Si la respuesta es "Sábado" o "Domingo", muestra "**¡Buen fin de semana!**". De lo contrario, muestra "**¡Buena semana!**".

```
JS app.js > ...
1 let diaDeLaSemana = prompt ("Ingresa el día de la semana");
2 if (diaDeLaSemana=="Sábado" || diaDeLaSemana== "Domingo"){
3     alert ("Buen fin de semana");
4 } else{
5     alert("Buena semana");
6 }
7 console.log (diaDeLaSemana);
```

- b. Verifica si un número ingresado por el usuario es positivo o negativo. Muestra una alerta informativa.

```
JS app.js > ...
1 let numero = prompt("Pon un número y te diré si es positivo o negativo");
2 if (numero>0){
3     alert ('Este numero es positivo ${numero}`);
4 } else {
5     alert ('Este numero es negativo ${numero}`);
6 }
7 console.log(numero);
```

```
JS app.js > ...
1 let numero = prompt("Pon un número y te diré si es positivo o negativo");
2 if (numero>0){
3     alert ('Este numero es positivo ${numero}`);
4 } else {
5     alert ('Este numero es negativo ${numero}`);
6 }
7 console.log(numero);
```

- c. Crea un sistema de puntuación para un juego. Si la puntuación es mayor o igual a 100, muestra "**¡Felicitaciones, has ganado!**". En caso contrario, muestra "**Intentalo nuevamente para ganar.**".
- d. Crea un mensaje que informe al usuario sobre el saldo de su cuenta, utilizando un template string para incluir el valor del saldo.
- e. Pide al usuario que ingrese su nombre mediante un `prompt`. Luego, muestra una alerta de bienvenida usando ese nombre.

The screenshot shows a code editor with the file `app.js` open. The code is as follows:

```

JS app.js > ...
1  let nombre = prompt("Antes de iniciar, indícame tu nombre");
2  alert(`Hola ${nombre}`);
3  console.log(nombre);
4

```

Below the code editor is a browser window displaying the output of the script. The title bar says "name". The main content area shows:

Esta página dice
Hola Amaris

Aceptar

10. Para saber más: punto y coma en JavaScript

En JavaScript, el uso del punto y coma (;) es una práctica recomendada. El lenguaje tiene un mecanismo llamado "inserción automática de punto y coma" (automatic semicolon insertion - ASI) que intenta agregar punto y comas automáticamente en ciertos puntos del código donde están ausentes.

Esto significa que, en algunos casos, JavaScript intentará "corregir" la falta de punto y coma insertándolo automáticamente. Sin embargo, la interpretación del ASI puede llevar a comportamientos inesperados y errores sutiles, especialmente cuando las reglas no son claras.

Por lo tanto, para evitar posibles problemas y garantizar la claridad del código, muchos desarrolladores prefieren agregar punto y coma de manera explícita en sus programas.

A pesar de que la inserción automática de punto y coma puede ayudar a mitigar errores de sintaxis, es una buena práctica agregar punto y coma manualmente para evitar ambigüedades y problemas de interpretación. Esto es especialmente importante en situaciones como cuando varias instrucciones están en la misma línea, al usar declaraciones de retorno de valor o al minimizar el código.

En proyectos colaborativos o de gran escala, la consistencia en el estilo de codificación y la claridad del código son cruciales, y el uso explícito de punto y coma contribuye a un código más legible y menos propenso a errores de interpretación por parte de los programadores y del propio mecanismo de ASI.

1. https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Lexical_grammar

11. Lo que aprendimos

En esta aula:

- Utilizamos la consola para probar y depurar nuestro código, mostrando mensajes y valores durante la ejecución del programa.
- Aprendimos a utilizar estructuras condicionales (`if/else`) para generar lógicas que permiten al programa tomar decisiones basadas en condiciones específicas.
- Implementamos un bloque de código para mostrar un mensaje en caso de que el intento del usuario no coincida con el número secreto.
- Usamos Template Strings para concatenar el número secreto con el valor almacenado en una variable y mostrar un mensaje personalizado.

Loops y tentativas

1. Proyecto del aula anterior

- a. Se ha descargado 2034-logica-programacion-1-Aula2_practicaDelJuego

2. Tips mayor o menor

```
JS app.js > ...
1  //Variables
2  let numeroSecreto = 2;
3  let numeroUsuario = prompt("Me indicas un número entre 1 y 10 por favor:");
4
5  console.log(numeroUsuario);
6  /*
7  Este código realiza
8  la comparación
9  */
10 if (numeroUsuario == numeroSecreto) {
11     //Acertamos, fue verdadera la condición
12     alert(`Acertaste, el número es: ${numeroUsuario}`);
13 } else {
14     //La condición no se cumplió
15     alert('Lo siento, no acertaste el número');
16 }
```

Hasta ahora, este es nuevo avance en el proyecto del juego. Pero necesitamos darle pistas al usuario para que pueda adivinar el número secreto. La idea es realizar una guía para poder encaminar la respuesta.

Para aportar con la ayuda, realizaremos condiciones anidadas, la cual es una condición dentro de otra condición.

```
JS app.js > ...
1  //Variables
2  let numeroSecreto = 5;
3  let numeroUsuario = prompt("Me indicas un número entre 1 y 10 por favor:");
4
5  console.log(numeroUsuario);
6  /*
7  Este código realiza
8  la comparación
9  */
10 if (numeroUsuario == numeroSecreto) {
11     //Acertamos, fue verdadera la condición
12     alert(`Acertaste, el número es: ${numeroUsuario}`);
13 } else {
14     // condición adicional para guiar la respuesta
15     if (numeroUsuario > numeroSecreto){
16         alert('El número secreto es menor');
17     } else
18         alert("El número secreto es mayor");
19     //La condición no se cumplió
20     alert('Lo siento, no acertaste el número');
21 }
```

3. Loops y bucles

- a. Con el punto anterior, pudimos dejar que el usuario pudiera jugar solo una vez, para que pueda volver a intentarlo sin tener que volver a recargar la página podemos agregar una acción para que lo intente otra vez.
- b. Las claves son: Hasta, Mientras o Para. Por el momento vamos a ocupar la palabra mientras, en JS será el comando WHILE. Para ocupar esta condición, debemos encerrar nuestro código bajo el

while con tabulación

```
JS app.js > ...
1  //Variables
2  let numeroSecreto = 5;
3  while(numeroUsuario != numeroSecreto){
4      let numeroUsuario = prompt("Me indicas un número entre 1 y 10 por favor:");
5
6      console.log(numeroUsuario);
7      /*
8       Este código realiza
9       la comparación
10      */
11     if (numeroUsuario == numeroSecreto) {
12         //Acertamos, fue verdadera la condición
13         alert(`Acertaste, el número es: ${numeroUsuario}`);
14     } else {
15         // condición adicional para guiar la respuesta
16         if (numeroUsuario > numeroSecreto){
17             alert('El número secreto es menor');
18         } else
19             alert("El número secreto es mayor");
20         //La condición no se cumplió
21         alert('Lo siento, no acertaste el número');
22     }
23 }
24 }
```

Con el WHILE estamos diciendo que mientras no se cumpla la condición, algo va a suceder en ese código.

!= significa diferente de

Pero tenemos un error, estamos declarando la variable dentro del repit y esto es un error. Para esto, tenemos que dejar la variable fuera del while . Sacamos la declaración de la variable.

Siempre debemos declarar las variables antes de cualquier acción.

Terminado de acomodar las variables y ubicar bien los {}, el código quedaría de la siguiente manera

```

JS app.js > ...
1  //Variables
2  let numeroSecreto = 5;
3  let numeroUsuario = 0;
4  while(numeroUsuario != numeroSecreto){
5      numeroUsuario = prompt("Me indicas un número entre 1 y 10 por favor:");
6
7      console.log(numeroUsuario);
8      /*
9      Este código realiza
10     la comparación
11     */
12     if (numeroUsuario == numeroSecreto) {
13         //Acertamos, fue verdadera la condición
14         alert(`Acertaste, el número es: ${numeroUsuario}`);
15     } else {
16         // condición adicional para guiar la respuesta
17         if (numeroUsuario > numeroSecreto){
18             alert('El número secreto es menor');
19         } else {
20             alert("El número secreto es mayor");
21         }
22         //La condición no se cumplió
23         alert('Lo siento, no acertaste el número');
24     }
25 }

```

4. Contador de intentos

- Para que nuestro juego tenga un límite de intento, aplicaremos un contador. Para esto crearemos una variable llamada intentos, este contador podemos dejarlo dentro del WHILE y le asignaremos un 1, ya que al menos tendrá 1 intento para iniciar en el juego.
- Si al primer intento lo adivino el numero, necesita incrementar los intentos, para esto JS tiene 3 formas de hacer un contador, la primera es la variable intentos + un intento mas.
- Para que el usuario pueda saber en cuantos intentos necesito para ganar, podemos completar mas la alerta de acertaste para que muestre el número de intentos.
- Otro punto a tener en cuenta, es que si el usuario acierta a la primera, el mensaje que le saldrá tendrá un problema lingüístico, ya que no concuerda "1 veces". Para esto vamos a agregar una variable para especificar cuando ocupar vez y veces. Aprovechamos el alert para adjuntar \${} y agregar la variable nueva.
- Pero ahora nos surge otro problema, necesitamos especificar que diferencie entre vez y veces, para esto nos ubicamos en el contador y ahí ubicamos la sentencia para que cuando lo realice en mas de 1 intentos, puede JS leer y deducir que la variable palabraVeces puede cambiar al activar el contador de intentos.

```

js app.js > ...
1 //Variables
2 let numeroSecreto = 5;
3 let numeroUsuario = 0;
4 let intentos = 1;
5 let palabraVeces = "vez";
6
7 while(numeroUsuario != numeroSecreto){
8     numeroUsuario = prompt("Me indicas un número entre 1 y 10 por favor:");
9
10    console.log(numeroUsuario);
11    /*
12     Este código realiza
13     la comparación
14     */
15    if (numeroUsuario == numeroSecreto) {
16        //Acertamos, fue verdadera la condición
17        alert(`Acertaste, el número es: ${numeroUsuario}. Lo hiciste en ${intentos} ${palabraVeces}`);
18    } else {
19        // condición adicional para guiar la respuesta
20        if (numeroUsuario > numeroSecreto){
21            alert('El número secreto es menor');
22        } else {
23            alert("El número secreto es mayor");
24        //La condición no se cumplió
25        alert('Lo siento, no acertaste el número');
26        }
27        //Contador de intentos
28        intentos = intentos + 1;
29        palabraVeces = "veces";| e
30    }
31 }
32

```

c d

5. Contador 1

Saber cómo trabajar con bucles de repetición, como el 'while', es esencial en la programación. Los bucles permiten automatizar tareas repetitivas y manejar grandes cantidades de datos de manera eficiente.

Sin embargo, es importante utilizarlos con precaución para evitar bucles infinitos (cuando la condición nunca se vuelve falsa, lo que puede bloquear el programa).

Siempre se recomienda tener una lógica que eventualmente haga que la condición se vuelva falsa para que el bucle termine correctamente.

A continuación, tenemos un ejemplo de código:

```
let contador = 1;

while (contador < 4) {
    console.log('Ejecutando la iteración ' + contador);
    contador = contador + 1;
}
```

[COPIA EL CÓDIGO](#)

Al ejecutar código, podemos afirmar que:

	La instrucción 'while (contador < 4)' inicia el bucle 'while'. Este continuará repitiendo el bloque de código dentro de las llaves {} mientras la condición 'contador < 4' sea verdadera. ¡Exactamente! Esa condición significa que el bucle continuará mientras el valor de la variable 'contador' sea menor que 4.
	El resultado final será la impresión de los mensajes "Ejecutando la iteración 1", "Ejecutando la iteración 2" y "Ejecutando la iteración 3" en la consola. ¡Exacto! El bucle se ejecutó tres veces, y el valor final de la variable 'contador' es 4. En la cuarta iteración, el valor de 'contador' es 4. En ese momento, la condición 'contador < 4' se vuelve falsa, ya que 4 no es menor que 4. El bucle deja de ejecutar el código dentro de las llaves y termina.
	C El resultado final será la impresión de los mensajes "Ejecutando la iteración 1", "Ejecutando la iteración 2", "Ejecutando la iteración 3" y "Ejecutando la iteración 4" en la consola.
	D Se mostrará un error en la consola indicando que el contador no está definido.

6. Bucle infinito

Willian está comenzando en el mundo de la programación y recientemente descubrió que existe una manera de ejecutar un bloque de comandos repetidamente mientras una condición preestablecida no se cumpla.

Animado por las posibilidades, decidió entrenar el uso de la estructura 'while()' en un proyecto personal de cálculo de medias aritméticas. Sin embargo, se encontró con un bucle infinito y no pudo descubrir por qué.

Sabiendo que tú también estás estudiando lógica de programación, Willian te pidió ayuda para localizar dónde está el error en su código:

```
let cantidadNumeros = prompt('Ingrese la cantidad de números para el cálculo de');
let suma = 0;
let contador = cantidadNumeros;

while(contador > 0){
    let numero = parseInt(prompt('Ingrese el numero:'));
    suma += numero;
}

let promedio = suma / cantidadNumeros;

console.log(promedio);
```

[COPIA EL CÓDIGO](#)

¿Qué cambio podrías sugerirle a Willian para que el código obtenga el resultado esperado? Elija la opción correcta:

A Cuando utilizamos bucles con contadores, debemos asegurarnos de que en algún momento alcancen la condición de parada del bucle. En el código anterior, el contador no tiene su valor alterado y para resolver el bucle infinito, simplemente debes incrementarlo en cada iteración, agregando la línea 'contador++' dentro del bucle.

De hecho, es importante modificar el valor del contador para que alcance la condición de parada. Sin embargo, al incrementar el contador en este ejemplo, siempre tomará valores mayores que 0, es decir, nunca alcanzará la condición de parada y el bucle infinito continuará.

B El código presenta un bucle infinito porque la condición de parada no se ha definido correctamente. Para que el programa funcione como se espera, es necesario cambiar el código de manera que:

```
while(contador > cantidadDeNumeros){
    //Código omitido
}
```

De hecho, la condición de parada se ha definido correctamente, pero el contador necesita ser decrementado en cada iteración para alcanzarla. Cuando definimos esta nueva condición de parada y no modificamos el valor de ninguna de estas variables, el bucle infinito persiste.

C El código presenta un bucle infinito porque las variables no alcanzan el valor de la condición de parada del bucle. Para resolver esto, simplemente debes agregar la línea 'cantidadDeNumeros++' dentro del bucle.

Al cambiar el valor de la variable cantidadDeNumeros, en realidad no estamos modificando el comportamiento del bucle; la forma correcta de detener el bucle es decrementar la variable contador en cada iteración.

 Cuando utilizamos bucles con contadores, debemos asegurarnos de que en algún momento alcancen la condición de parada del bucle. En el código anterior, el contador no tiene su valor alterado, y para resolver el bucle infinito, simplemente debes decrementarlo en cada iteración, agregando la línea 'contador--' dentro del bucle.

Al decrementar el valor de la variable contador en cada iteración, eventualmente será menor o igual a 0, lo que detendrá la ejecución del bucle.

7. Desafío: Hora de practicar

Hemos llegado a otra lista de actividades (no obligatorias) para que practiques y refuerces aún más tu aprendizaje. ¿Vamos a hacerlo?

1. Crea un contador que comience en 1 y vaya hasta 10 usando un bucle 'while'. Muestra cada número.
2. Crea un contador que comience en 10 y vaya hasta 0 usando un bucle 'while'. Muestra cada número.
3. Crea un programa de cuenta regresiva. Pide un número y cuenta desde 0 hasta ese número utilizando un bucle 'while' en la consola del navegador.
4. Crea un programa de cuenta progresiva. Pide un número y cuenta desde 0 hasta ese número utilizando un bucle 'while' en la consola del navegador.

8. Para saber más: operadores lógicos

Cuando escribimos programas en JavaScript, a menudo nos encontramos con la necesidad de tomar decisiones basadas en condiciones. Es aquí donde los operadores lógicos entran en escena y nos ayudan a crear una lógica sólida y eficaz.

A continuación, vamos a explorar los operadores lógicos de una manera simple y fácil de entender. Tendremos ejemplos claros para ilustrar su funcionamiento.

AND (&&)

El operador AND, representado por el símbolo "&&", se utiliza para combinar dos condiciones y evaluar si ambas son verdaderas. Si ambas condiciones son verdaderas, el resultado será... verdadero. De lo contrario, lógicamente será falso. Por ejemplo:

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

// si la edad es mayor de 18 y tiene una licencia...
if (edad > 18 && tieneLicencia) {
    console.log("Puede conducir!");
} else {
    console.log("No puede conducir!");
}
```

[COPIA EL CÓDIGO](#)

OR(||)

El operador OR, representado por los símbolos "||", se utiliza para verificar si al menos una de las condiciones es verdadera. Si una de las condiciones es verdadera, el resultado será verdadero. Si ambas son falsas, el resultado será falso. Aquí tienes un ejemplo: let tieneManzana = false; let tieneBanana = true;

```
// si tiene manzana o tiene banana...
if (tieneManzana || tieneBanana) {
    console.log("Tienes frutas!");
} else {
    console.log("No tienes frutas.");
}
```

[COPIA EL CÓDIGO](#)

Otros tipos de operadores lógicos

Operador	Nombre	Ejemplo	Resultado
==	Igual	A == B	Verdadero si A es igual a B
!=	Diferente	A != B	Verdadero si A no es igual a B
<	Menor que	A < B	Verdadero si A es menor que B
>	Mayor que	A > B	Verdadero si A es mayor que B
<=	Menor o igual	A <= B	Verdadero si A es menor o igual a B
>=	Mayor o igual	A >= B	Verdadero si A es mayor o igual a B

Operadores Lógicos

Operador	Nombre	Ejemplo	Resultado
&&	Y / AND	(A > B) && (B == C)	Verdadero si A es mayor que B y B es igual a C
	O / OR	(A > B) (B == C)	Verdadero si A es mayor que B o B es igual a C
!	NEGACIÓN	!(A == B)	Verdadero si A NO es igual a B

<https://www.aluracursos.com/blog/como-utilizar-operadores-de-comparacion-en-javascript>

9. Lo que aprendimos

En esta lección:

- Aprendimos a comprobar si un número es mayor o menor que otro usando estructuras condicionales (if/else), y sobre condicionales anidados, en nuestro programa.
- Usamos el bucle " while " para repetir un bloque de código mientras una cierta condición sea verdadera, permitiendo así que el programa realice una acción varias veces.
- Implementamos un contador de intentos para hacer un seguimiento y mostrar la cantidad de veces que el usuario intentó adivinar un número secreto. Podemos hacer esto, por ejemplo, en un juego de adivinanza.

Buenas prácticas en programación.

1. Proyecto anterior

- a. Descargar <https://github.com/alura-es-cursos/2034-logica-programacion-1/tree/Aula3>

2. Break

- a. Vamos a subir la dificultad en nuestro juego limitando los intentos, para esto aprenderemos un nuevo concepto llamado "la ruptura forzada" o "salir forzado" de un bucle.
- b. Agregamos una condición extra abajo del comando de los intentos, esta condición se realizará con IF, podemos también adjuntar una alerta donde le informe al usuario que ha llegado al límites de intentos. Al final de este IF, agregaremos el BREAK, esto le dará a entender a JS que al intentarlo x número de veces deberá romper el bucle.

```
JS app.js > ...
1  //Variables
2  let numeroSecreto = 5;
3  let numeroUsuario = 0;
4  let intentos = 1;
5  let palabraVezes = "vez";
6
7  while(numeroUsuario != numeroSecreto){
8      numeroUsuario = prompt("Me indicas un número entre 1 y 10 por favor:");
9
10     console.log(numeroUsuario);
11     /*
12     Este código realiza
13     la comparación
14     */
15     if (numeroUsuario == numeroSecreto) {
16         //Acertamos, fue verdadera la condición
17         alert(`Acertaste, el número es: ${numeroUsuario}. Lo hiciste en ${intentos} ${palabraVezes}`);
18     } else {
19         // condición adicional para guiar la respuesta
20         if (numeroUsuario > numeroSecreto){
21             alert('El número secreto es menor');
22         } else {
23             alert("El número secreto es mayor");
24             //La condición no se cumplió
25             alert('Lo siento, no acertaste el número');
26         }
27         //Contador de intentos
28         intentos = intentos + 1;
29         palabraVezes = "veces";
30         if (intentos > 3) {
31             alert ("Haz llegado al máximo de intentos");
32             break;
33         }
34     }
35 }
```

3. Solo 5 intentos

Pregunta: ¿Cómo implementarás un límite de 5 intentos para que los clientes adivinen el número secreto?

A

Utilizando un bucle while sin la instrucción break .

```
let numeroSecreto = Math.floor(Math.random() * 10);
let intentos = 0;
while (intentos < 5) {
  let intento = parseInt(prompt("Ingrese un número del 0 al 9:"));
  intentos++;
  if (intento == numeroSecreto) {
    alert(`¡Eso es! Descubriste el número secreto ${numeroSecreto}`);
  }
}
```



Utilizando un bucle while con la instrucción break cuando el cliente adivine el número.

```
let numeroSecreto = Math.floor(Math.random() * 10);
let intentos = 0;
while (intentos < 5) {
  let intento = parseInt(prompt("Ingrese un número del 0 al 9:"));
  intentos++;
  if (intento == numeroSecreto) {
    alert(`¡Eso es! Descubriste el número secreto ${numeroSecreto}`);
    break;
  }
}
```

Este código implementa correctamente un límite de 5 intentos y también utiliza la instrucción break cuando se adivina el número secreto.

C

Utilizando un bucle for sin la instrucción break .

```
let numeroSecreto = Math.floor(Math.random() * 10);
for (let intentos = 0; intentos < 5; intentos++) {
  let intento = parseInt(prompt("Ingrese un número del 0 al 9:"));
  if (intento == numeroSecreto) {
    alert(`¡Eso es! Descubriste el número secreto ${numeroSecreto}`);
  }
}
```



Usando un bucle for junto con la instrucción break cuando el cliente adivine el número o cuando el número de intentos llegue a 5.

```
let numeroSecreto = Math.floor(Math.random() * 10);
for (let intentos = 0; intentos < 5; intentos++) {
  let intento = parseInt(prompt("Ingrese un número del 0 al 9:"));
  if (intento == numeroSecreto) {
    alert(`¡Eso es! Descubriste el número secreto ${numeroSecreto}`);
    break;
  }
}
```

Este código implementa correctamente un límite de 5 intentos. Utiliza un bucle for y finaliza el bucle cuando se adivina el número secreto o cuando se alcanzan 5 intentos.

4. Operador ternario

- Dejar condiciones con valores literales no es una buena práctica, ya que complica la mantenibilidad y el crecimiento de nuestro programa. Entonces, vamos a definir una variable llamada `maximosIntentos`. Por ahora, manteniendo la funcionalidad, le asignamos el valor de 3 y hacemos el cambio de que si los intentos del usuario son mayores que `maximosIntentos`.
- Realizaremos un cambio de optimización con un par de recursos, cambiaremos la forma de hacer el contador, existe una forma mas abreviada, pueden ser: `intentos +=1;` ó `intentos++;` Esto nos ayudara a reducir el código. Aquí lo que cambiamos fue el incrementados intentos.
- Nuestra segunda mejora será el “operador ternario”, al usar template strings dentro de nuestra frase “Acertaste, el número...” para usar las palabras en singular o plural en función de intentos, resumimos todo en una sola frase. Lo que se esta haciendo en este cambio es que en la

condición `intentos == 1`? le estamos diciendo que si el valor de la variable `intentos` es igual a uno es verdadero (ya que el usuario adivino con un solo intento), por ende debe mostrar en el mensaje "vez", pero si es mas de un intento, esto deberá ser falso. (: veces)

`==` comparación

? si la condición es verdadera, usa lo de la izquierda de los dos puntos :

: (simboliza `else` ó si no)si la condición es falsa, usa lo de la derecha

```
js app.js > ...
1 //Variables
2 let numeroSecreto = 5;
3 let numeroUsuario = 0;
4 let intentos = 1;
5 //let palabraVezes = "vez"; C
6 let maximosIntentos = 3;
7
8 while(numeroUsuario != numeroSecreto){
9     numeroUsuario = prompt("Me indicas un número entre 1 y 10 por favor:");
10
11    console.log(numeroUsuario);
12    /*
13     Este código realiza
14     la comparación
15     */
16    if (numeroUsuario == numeroSecreto) {
17        //Acertaste, fue verdadera la condición
18        alert(`Acertaste, el número es: ${numeroUsuario}, Lo hiciste en ${intentos} ${intentos == 1 ? 'vez': "veces"}`);
19    } else {
20        // condición adicional para guiar la respuesta
21        if (numeroUsuario > numeroSecreto){
22            alert('El número secreto es menor');
23        } else {
24            alert("El número secreto es mayor");
25        }
26        //Contador de intentos
27        //intentos = intentos + 1;
28        intentos++;
29        //palabraVezes = "veces";
30        if (intentos > maximosIntentos) {
31            alert ('Haz llegado al máximo de ${maximosIntentos} intentos');
32            break;
33        }
34    }
35}
36 }
```

Un operador ternario es una forma corta de escribir una estructura `if ... else`. Esta tiene su propia sintaxis:

```
condición ? valor_si_verdadero : valor_si_falso
```

5. Refactorizando

Sabiendo que tú has estudiado sobre el operador recientemente, ella te pidió ayuda y te mostró el código que quiere refactorizar:

```
let palabraPersona = "";

if(cantidadPersonas == 1){
    palabraPersona = "persona";
} else{
    palabraPersona = "personas"
}
```

[COPIA EL CÓDIGO](#)

¿Cómo podrías mostrarle a Aline un ejemplo de cómo transformar este fragmento de código en otro que tenga el mismo comportamiento, pero utilice el operador ternario?

A	<pre>let palabraPersona = if(cantidadPersonas == 1) ? "persona" : "personas";</pre>	
B	<pre>let palabraPersona = cantidadPersonas == 1 ? "persona" ; "personas";</pre>	
C	<pre>let palabraPersona = cantidadPersonas == 1 ? "persona" : "personas";</pre>	
D	<pre>let palabraPersona = cantidadPersonas ? "persona" : "personas";</pre>	

Con la utilización de este código con el operador ternario, Aline podrá reemplazar varias líneas de código con solo una.

6. Math.random () en nuestro código

- https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/Math/random
- Lo que nos falta ahora en nuestro juego es que el número secreto sea aleatorio. La función Math. tiene una amplia funciones para trabajar, pero ahora ocuparemos Math.random.
- Por ejemplo, si ocuparemos Math.random la consola solo nos arrojara una infinidad de número random, pero podemos limitarlos con ()*10, esto quiere decir, que limitaremos de 0 a 10 número. El problema aqui es que al no estar especificado, nos arrojara hasta los numero decimales (0.45, 2.487, 9.99, etc) Para esto ocuparemos las otras funciones que ofrece Math.
- Al ocupar Math.floor(Math.random() estamos pidiendo que nos de un numero ENTERO (floor) aleatorio. Pero para ser aun mas específicos, podemos solicitar lo siguiente:
Math.floor(Math.random() *10); aquí estamos solicitando un número aleatorio entero del 0 al 9
- Para poder incluir el número 10 y excluir el 0, simplemente a Math.floor(Math.random() *10) debemos agregar un +1 Math.floor(Math.random() *10) +1;

```
> Math.random();
< 0.7968641927515362
> Math.random()*10;
< 4.588499635008492
> Math.floor(Math.random()*10);
< 7
> Math.floor(Math.random()*10)+1;
< 4
> Math.floor(Math.random()*10)+1;
< 5
```

Esto se realizó en la consola de x página web

7. Número aleatorio

La función Math.random en JavaScript genera un número decimal aleatorio entre 0 (inclusivo) y 1 (exclusivo) de forma pseudoaleatoria. Esto significa que los números generados pueden estar en cualquier lugar entre 0 (inclusivo) y casi 1 (exclusivo), con una precisión de hasta 16 decimales. La función Math.random utiliza un valor interno inicial que generalmente se basa en la hora actual del sistema, generando números pseudoaleatorios.

A continuación, se muestran algunos ejemplos:

```
0.5248738910328501
0.08458620904957355
0.9347284927568912
```

[COPIA EL CÓDIGO](#)

Teniendo esto en cuenta, analiza las siguientes opciones y marca solo la verdadera:

A

Para generar un número entre 1 y 3, podemos usar el código: `let numeroAleatorio1a3 = parseInt(Math.random() * 3);`

⊗



Para generar un número entre 1 y 3, podemos usar el código: `let numeroAleatorio1a3 = parseInt(Math.random() * 3) + 1;`

⊗

El código anterior genera un número entero aleatorio entre 1 y 3, ambos inclusive. Por lo tanto, los posibles números que pueden generarse con este código son: 1, 2 y 3.

C

Para generar un número entre 1 y 3, podemos usar el código: `let numeroAleatorio1a3 = parseInt(Math.random() * 4);`

⊗

8. Math.random () en nuestro código

- a. Hemos aplicado a nuestro código lo aprendido con Math.random, cambiando la variable numeroSecreto = 5 por lo aprendido y además para verificar nuestra escritura, agregamos un console.log para ir testeando.

```
JS app.js > ...
1 //Variables
2 let numeroSecreto = Math.floor(Math.random()*10)+1;
3 let numeroUsuario = 0;
4 let intentos = 1;
//let palabraVceces = "vez";
6 let maximosIntentos = 3;
7
8 console.log(numeroSecreto);
9
10 while(numeroUsuario != numeroSecreto){
11   numeroUsuario = prompt("Me indicas un número entre 1 y 10 por favor:");
12
13   console.log(numeroUsuario);
14
15   if (numeroUsuario == numeroSecreto) {
16     //Acertamos, fue verdadera la condición
17     alert(`Acertaste, el número es: ${numeroUsuario}. Lo hiciste en ${intentos} ${intentos == 1 ? 'vez': "veces"} `);
18   } else {
19     // condición adicional para guiar la respuesta
20     if (numeroUsuario > numeroSecreto){
21       alert('El número secreto es menor');
22     } else {
23       alert("El número secreto es mayor");
24     }
25     //Contador de intentos
26     //intentos = intentos + 1;
27     intentos++;
28     //palabraVceces = "veces";
29     if (intentos > maximosIntentos) {
30       alert(`Haz llegado al máximo de ${maximosIntentos} intentos`);
31       break;
32     }
33   }
34 }
35 }
```

Para métodos de jugabilidad, borraremos el console.log(numeroSecreto);

- b. En el prompt ("Me indicas un n...) nos retorna el número del usuario y es posible saber el tipo de dato con la funcionalidad llamada TYPEOF (Te dice de que tipo es un valor o variable)

¿Por que es importante usar TYPEOF?

Porque evita errores de lógica, verifica qué tipo de dato te están devolviendo y depura el código.

```
typeof "hola"      // "string"
typeof 42          // "number"
typeof true         // "boolean"
typeof undefined    // "undefined"
typeof null         // "object" (esto es un detalle raro de JavaScript)
typeof NaN          // "number"
```

- c. También podemos usar el PARSEINT (su función es convertir un texto (string) en número entero)
Ejemplo:

```
parseInt("7"); // Devuelve el número 7
parseInt("42"); // Devuelve el número 42
parseInt("5.8"); // Devuelve el número 5 (solo la parte entera)
parseInt("abc"); // Devuelve NaN (Not a Number, porque no puede convertir letras)
```

¿Por que es importante usar PARSEINT?

Porque cuando trabajamos con `prompt`, siempre recibimos texto. Si queremos hacer comparaciones numéricas, sumas o cálculos, debemos asegurarnos de que el valor sea un número real, no texto.

```

js app.js > ...
1 //Variables
2 let numeroSecreto = Math.floor(Math.random()*10)+1;
3 let numeroUsuario = 0;
4 let intentos = 1;
5 //let palabraVeces = "vez";
6 let maximosIntentos = 3;
7
8 console.log(numeroSecreto);
9
10 while(numeroUsuario != numeroSecreto){
11     C numeroUsuario = parseInt(prompt("Me indicas un número entre 1 y 10 por favor:"));
12     D console.log(typeof(numeroUsuario));
13
14
15     if (numeroUsuario == numeroSecreto) {
16         //Acertamos, fue verdadera la condición
17         alert(`Acertaste, el número es: ${numeroUsuario}. Lo hiciste en ${intentos} ${intentos == 1 ? 'vez': "veces"}`);
18     } else {
19         // condición adicional para guiar la respuesta
20         if (numeroUsuario > numeroSecreto){
21             alert('El número secreto es menor');
22         } else {
23             alert("El número secreto es mayor");
24         }
25     }
26     //contador de intentos
27     //intentos = intentos + 1;
28     intentos++;
29     //palabraVeces = "veces";
30     if (intentos > maximosIntentos) {
31         alert(`Haz llegado al máximo de ${maximosIntentos} intentos`);
32         break;
33     }
34 }
35

```

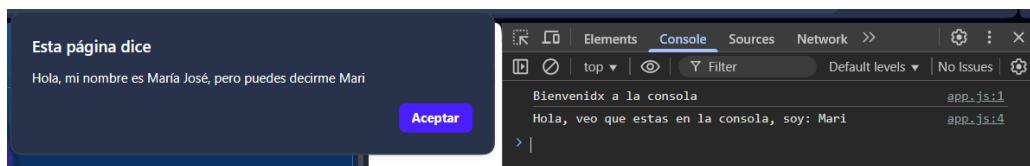
9. Desafío: Hora de practicar

- Crea un programa que utilice `console.log` para mostrar un mensaje de bienvenida.
- Crea una variable llamada "nombre" y ásñale tu nombre. Luego, utiliza `console.log` para mostrar el mensaje "¡Hola, [tu nombre]!" en la consola del navegador.
- Crea una variable llamada "nombre" y ásñale tu nombre. Luego, utiliza `alert` para mostrar el mensaje "¡Hola, [tu nombre]!".

```

1 console.log("Bienvenidx a la consola");
2
3 let nombre = "Mari";
4 console.log(`Hola, veo que estas en la consola, soy: ${nombre}`);
5 alert(`Hola, mi nombre es María José, pero puedes decirme ${nombre}`);

```

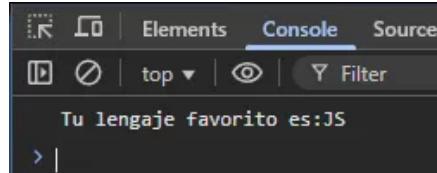
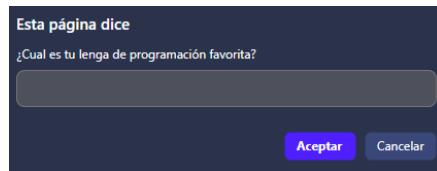


- Utiliza `prompt` y haz la siguiente pregunta: ¿Cuál es el lenguaje de programación que más te gusta?. Luego, almacena la respuesta en una variable y muestra la respuesta en la consola del navegador.

```

js app.js > ...
1 let lenguaDeProgramacion = prompt ("¿Cuál es tu lengua de programación favorita?");
2 console.log("Tu lenguaje favorito es:" + lenguaDeProgramacion);
3

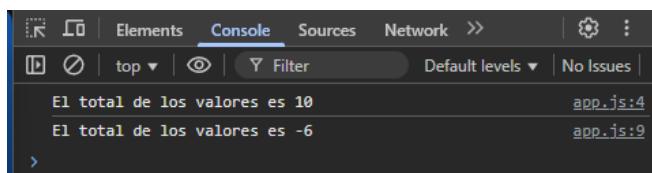
```



RECORDAR: el signo + en JS (cuando se usa con cadena de texto) sirve para unir texto. Eso se llama CONCATENAR. Entonces en el ejemplo "Tu lenguaje favorito es: " es un texto, mientras lenguajeDeProgramación es una variable que guarda lo que el usuario escribió. El + une ambos, para que se muestre todo junto. Para ahorrar problemas, recuerda usar los amados TEMPLATE ` blablabla \${.....}`

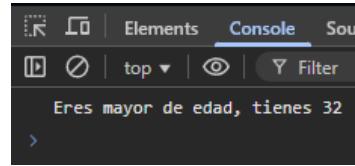
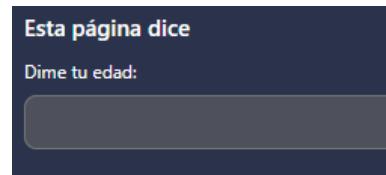
- e. Crea una variable llamada "valor1" y otra llamada "valor2", asignándoles valores numéricos de tu elección. Luego, realiza la suma de estos dos valores y almacena el resultado en una tercera variable llamada "resultado". Utiliza `console.log` para mostrar el mensaje "La suma de [valor1] y [valor2] es igual a [resultado]." en la consola.
- f. Crea una variable llamada "valor1" y otra llamada "valor2", asignándoles valores numéricos de tu elección. Luego, realiza la resta de estos dos valores y almacena el resultado en una tercera variable llamada "resultado". Utiliza `console.log` para mostrar el mensaje "La diferencia entre [valor1] y [valor2] es igual a [resultado]." en la consola.

```
JS app.js > ...
1 let valor1 = 2;
2 let valor2 = 8;
3 let resultado = valor1+valor2;
4 console.log(`El total de los valores es ${resultado}`);
5
6 let valor3 = 2;
7 let valor4 = 8;
8 let resultadoResta = valor3-valor4;
9 console.log(`El total de los valores es ${resultadoResta}`);
```



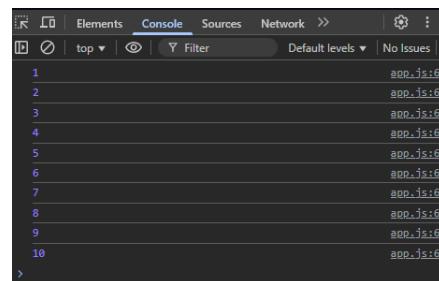
- g. Pide al usuario que ingrese su edad con `prompt`. Con base en la edad ingresada, utiliza un `if` para verificar si la persona es mayor o menor de edad y muestra un mensaje apropiado en la consola.
- h. Crea una variable "numero" y solicita un valor con `prompt`. Luego, verifica si es positivo, negativo o cero utilizando un `if-else` y muestra el mensaje correspondiente.

```
JS app.js > ...
1 let edadUsuario = prompt("Dime tu edad:");
2 if (edadUsuario>=18){
3   console.log(`Eres mayor de edad, tienes ${edadUsuario}`);
4 } else {
5   console.log(`Eres menor de edad, tienes ${edadUsuario}`);
6 }
```



- i. Utiliza un bucle `while` para mostrar los números del 1 al 10 en la consola.

```
JS app.js > ...
1
2 let numero = 1;
3 while (numero <= 10) {
4   console.log(numero);
5   numero++;
6 }
```



RECORDAR:

Los paréntesis llave {} definen bloques de códigos (forman la caja), es decir, ayuda a agrupar el conjunto de instrucciones que se ejecutan si la condición (caja) se cumple.

El ++ es un operador de incremento, ósea decir: contador = contador +1 es lo mismo que decir contador++, pero el ultimo ejemplo es una buena practica a utilizar.

- j. Crea una variable "nota" y asígnale un valor numérico. Utiliza un `if-else` para determinar si la nota es mayor o igual a 7 y muestra "Aprobado" o "Reprobado" en la consola.

```
let nota = prompt("Escribe tu nota");

if (nota >= 7) {
  alert(`Aprobaste con una nota ${nota}`);
  console.log(`Aprobaste con una nota ${nota}`);
} else {
  alert(`Reprobaste con una nota ${nota}`);
  console.log(`Reprobaste con una nota ${nota}`)
}
```

- k. Utiliza `Math.random` para generar cualquier número aleatorio y muestra ese número en la consola.
l. Utiliza `Math.random` para generar un número entero entre 1 y 10 y muestra ese número en la consola.

m. Utiliza `Math.random()` para generar un número entero entre 1 y 1000 y muestra ese número en la consola.

```
> Math.random()
< 0.5669393114654494
> Math.random()*10
< 3.9368814583932163
> Math.floor(Math.random()*10)
< 2
> Math.floor(Math.random()*10)+1
< 1
> Math.floor(Math.random()*10)+1
< 9
> Math.floor(Math.random()*10)+1
< 2
> Math.floor(Math.random()*100)+1
< 9
> Math.floor(Math.random()*100)+1
< 93
> Math.floor(Math.random()*1000)+1
< 369
> Math.floor(Math.random()*10000)+1
< 1497
```

10. Para saber más: ¿Necesito memorizar cada línea de código o comando?

Comprender cada línea de código o comando en detalle es ciertamente una aspiración loable, pero no es necesario memorizar. El desarrollo de software moderno es una tarea compleja, y los lenguajes de programación ofrecen una amplia gama de recursos y bibliotecas.

En lugar de memorizar cada línea, es más valioso entender los conceptos fundamentales detrás de las estructuras de programación y saber cómo utilizar la documentación de manera efectiva.

La documentación de un lenguaje de programación es una herramienta esencial para todos los desarrolladores. No solo proporciona una referencia rápida para la sintaxis y los comandos, sino que también explica los conceptos subyacentes, ofrece ejemplos prácticos y ayuda a comprender los diferentes recursos disponibles.

A través de la documentación, puedes aprender a usar bibliotecas, explorar casos de uso avanzados y comprender las mejores prácticas recomendadas por la comunidad. Esto ahorra tiempo, evita errores y te permite mantenerte actualizado con las últimas actualizaciones del lenguaje.

En lugar de preocuparte por memorizar cada detalle, concéntrate en desarrollar habilidades de resolución de problemas, comprender los principios de diseño de software y aprender a investigar de manera eficiente en la documentación. La capacidad de leer e interpretar la documentación es una habilidad valiosa, ya que te permite aprender nuevos lenguajes y tecnologías de manera efectiva, adaptándote rápidamente a los cambios en el panorama de desarrollo. Por lo tanto, en tu viaje como desarrollador, recuerda que la habilidad de comprender y usar la documentación es tan importante como saber escribir código.

En esta lección, vimos cómo usar la documentación para generar un número aleatorio [a través de la documentación de Mozilla](#). Sin embargo, también existe W3Schools (W3S), que es un recurso en línea ampliamente conocido y utilizado para aprender diversas tecnologías de desarrollo web, incluido JavaScript (JS). El sitio web ofrece tutoriales interactivos, ejemplos de código, referencias de sintaxis y conceptos fundamentales relacionados con JavaScript y otros lenguajes web.

Al explorar W3Schools, puedes adquirir una comprensión sólida de los principios de JS, desde lo básico hasta temas más avanzados, como la manipulación del DOM, interacciones con el usuario y solicitudes asíncronas. Es una herramienta valiosa para mejorar tus habilidades en JavaScript, ofreciendo un entorno práctico para experimentar con código, comprender conceptos clave y aplicar el conocimiento adquirido en tus propios proyectos de desarrollo web.

11. Lo que aprendimos

En esta lección:

- Aprendimos a evitar código duplicado utilizando estrategias para mostrar la palabra “intento” en singular o plural, dependiendo del número de intentos realizados.
- Analizamos la documentación y encontramos información relevante para el desarrollo de nuestro programa.
- Descubrimos cómo emplear la función `Math.random()` para generar un número aleatorio, lo que permite la creación de desafíos y juegos más dinámicos y variados.

Desafío

1. Proyecto de aula anterior

- Descargar proyecto: <https://github.com/alura-es-cursos/2034-logica-programacion-1/tree/Aula4>

2. Desafío

- Implementa nuevas funciones en tu código

- Una de las funciones que siento que falta en el juego, es seleccionar la dificultad. Dividir el juego por niveles, se puede abrir ventanas para diversidad de jugadores, desde edades mas pequeñas, nivel de tolerancia para cada usuario o simplemente ir subiendo o bajando la dificultad según lo requiera el usuario. Para esto investigue si existe la posibilidad de hacerlo y encontré una abanico de herramientas que me pueden ayudar a plasmar mi idea. Estas son:

- i. switch: Estructura de control, es como if, pero me permite comparar una misma variable contra varios valores posibles (casos). Se utiliza para simplificar múltiples comparaciones cuando una misma variable puede tener varios valores distintos.

La ocuparemos para entregarle a nuestro juego los niveles de dificultades (fácil, medio y difícil)

- ii. case: Define una posible opción o valor dentro de un switch. Cuando usamos switch, estamos revisando una sola variable para ver cuál case coincide. Se utiliza cuando estamos evaluando una misma variable con varias opciones posibles. El case se usa solo cuando ocupamos switch

Con case, crearemos los niveles de dificultad y le pondremos un límites de posibles números secretos y un límite de intentos. También en cada case, cerraremos con un break para romper el bucle

- iii. toLowerCase: es un método que aplica a cadenas de texto (string) y convierte todas las letras en minúsculas. Se utiliza para evitar errores por mayúsculas o minúsculas cuando comparas texto.

En nuestro juego, ocuparemos este método para que el usuario pueda escribir su nivel de dificultad. JS es muy sensible a los string con mayúsculas y minúsculas, entonces al usarlo en nuestro código, nos ahorraremos errores por tipo (FACÍL, Fácil, fAcíl, etc)

- iv. Para terminar de limitar bien los límites de cada nivel, crearemos una variable llamada limiteSuperior, esta variable nos ayudara a poner a trazar el límite de las dificultades para cada elección realizada por el usuario y esta la adjuntaremos dentro de la la caja de case según la dificultades con sus máximos límites de intentos. Realizado esto, debemos tener presente en cambiar la variable Math.random () *10 a :(Math.random()*limiteSuperior) + 1;

- c. Por otro parte, leyendo algunos pdf, encontré una función llamada isNaN(¿esto NO es un número?). Esta función retorna true si el valor NO es un número y se utiliza para evitar errores y validar entradas.

- i. En nuestro juego usamos esta función porque estamos usando un prompt para pedirle al usuario un número, pero prompt() siempre devuelve un texto (string), luego usamos parseInt(...) para convertirlo a número, entonces si el usuario escribe algo que no es un número válido, parseInt () devuelve NaN y es aquí donde entra en juego isNaN(). Por ejemplo, si seleccionamos dificultad fácil, debemos escribir un numero entre 1 a 50, pero en el caso de escribir 55, nos aparecerá un mensaje que nos indicara que ingresar un número válido entre 1 y 50.

- d. Otra herramienta a utilizar es continue, esta es una palabra clave que se utiliza dentro de bucle (while, for, do...). Se utiliza en ciertos casos como para ignorar entradas invalidas, saltar iteraciones innecesarias y filtrar cierto valores

- i. En el juego utilizamos esta herramienta para evitar que cuente como intento una entrada inválida. Por ejemplo, si el usuario en ves de escribir un número solicitado según la dificultad (-5 o perro), el juego no ejecutara el resto del código, pero si volvera a realizar la pregunta para que ingrese un número válido.
- e. Para finalizar, para darle un toque mas llamativos, al principio del curso, se comentó que existía un lenguaje de emojis y se aprovecho la virtud de JS para agregarlos al código.
 - i. Podemos usar los emojis en las alerts, console.log, prompt, cadenas de texto normales ("' ''') e intefaces de HTML

```
// Elegir dificultad
let dificultad = prompt("Elige dificultad: fácil, medio o difícil").toLowerCase();
let limiteSuperior;
let maximosIntentos;

switch (dificultad) {
  case "fácil":
    limiteSuperior = 50;
    maximosIntentos = 10;
    break;
  case "medio":
    limiteSuperior = 100;
    maximosIntentos = 7;
    break;
  case "difícil":
    limiteSuperior = 500;
    maximosIntentos = 5;
    break;
  default:
    alert("Dificultad no válida. Se asignará dificultad media.");
    limiteSuperior = 100;
    maximosIntentos = 7;
    console.log(limiteSuperior);
}

// Variables principales
let numeroSecreto = Math.floor(Math.random() * limiteSuperior) + 1;
let numeroUsuario = 0;
let intentos = 1;

console.log("Número secreto:", numeroSecreto);

while (numeroUsuario != numeroSecreto) {
  numeroUsuario = parseInt(prompt(`Indica un número entre 1 y ${limiteSuperior}`));

  // Validación
  if (isNaN(numeroUsuario) || numeroUsuario < 1 || numeroUsuario > limiteSuperior) {
    alert(`Por favor, ingresa un número válido entre 1 y ${limiteSuperior}`);
    continue;
}
}
```

```

console.log(typeof numeroUsuario);

if (numeroUsuario == numeroSecreto) {
    alert('🎉 Acertaste, el número es: ${numeroUsuario}. Lo hiciste en ${intentos} ${intentos ==
1 ? 'vez' : 'veces'}');
} else {
    if (numeroUsuario > numeroSecreto) {
        alert('☒ El número secreto es menor');
    } else {
        alert('☑ El número secreto es mayor');
    }
}

intentos++;

if (intentos > maximosIntentos) {
    alert('✖ Has llegado al máximo de ${maximosIntentos} intentos. El número era: ${numero
Secreto}');
    break;
}
}
}

```

```

js app.js > ...
1 // Elegir dificultad
2 let dificultad = prompt("Elige dificultad: fácil, medio o difícil").toLowerCase();
3 let limiteSuperior;
4 let maximosIntentos;
5
6 switch (dificultad) { bi
7 bii case "fácil":
8     limiteSuperior = 50;
9     maximosIntentos = 10;
10    break;
11    case "medio":
12        limiteSuperior = 100;
13        maximosIntentos = 7;
14        break;
15    case "difícil":
16        limiteSuperior = 500;
17        maximosIntentos = 5;
18        break;
19    default:
20        alert("Dificultad no válida. Se asignará dificultad media.");
21        limiteSuperior = 100;
22        maximosIntentos = 7;
23        console.log(limiteSuperior);
24    }
25
26 // Variables principales

```

```

25 // Variables principales
26 let numeroSecreto = Math.floor(Math.random() * limiteSuperior) + 1;
27 let numeroUsuario = 0;
28 let intentos = 1;
29
30 console.log("Número secreto:", numeroSecreto);
31
32 while (numeroUsuario != numeroSecreto) {
33     numeroUsuario = parseInt(prompt(`Indica un número entre 1 y ${limiteSuperior}:`));
34
35     // Validación
36     if (isNaN(numeroUsuario) || numeroUsuario < 1 || numeroUsuario > limiteSuperior) {
37         alert(`Por favor, ingresa un número válido entre 1 y ${limiteSuperior}.`);
38         continue;
39     }
40
41     console.log(typeof numeroUsuario);
42
43     if (numeroUsuario == numeroSecreto) {
44         alert(`🎉 Acertaste, el número es: ${numeroUsuario}. Lo hiciste en ${intentos} ${intentos == 1 ? 'vez' : 'veces'}.`);
45     } else {
46         if (numeroUsuario > numeroSecreto) {
47             alert(`🔴 El número secreto es menor`);
48         } else {
49             alert(`🔴 El número secreto es mayor`);
50         }
51
52         intentos++;
53
54         if (intentos > maximosIntentos) {
55             alert(`❌ Has llegado al máximo de ${maximosIntentos} intentos. El número era: ${numeroSecreto}`);
56             break;
57         }
58     }
59 }
60

```

3. Solución del desafío

```

js app.js > ...
1 //Variables
2 let numeroSecreto = Math.floor(Math.random()*10)+1;
3 let numeroUsuario = 0;
4 let intentos = 1;
5 //let palabraVeces = "vez";
6 let maximosIntentos = 3;
7
8 console.log(numeroSecreto);
9
10 while(numeroUsuario != numeroSecreto){
11     numeroUsuario = parseInt(prompt("Me indicas un número entre 1 y 10 por favor:"));
12
13     console.log(typeof(numeroUsuario));
14
15     if (numeroUsuario == numeroSecreto) {
16         //Acertamos, fue verdadera la condición
17         alert(`Acertaste, el número es: ${numeroUsuario}. Lo hiciste en ${intentos} ${intentos == 1 ? 'vez': "veces"}`);
18     } else {
19         // condición adicional para guiar la respuesta
20         if (numeroUsuario > numeroSecreto){
21             alert('El número secreto es menor');
22         } else {
23             alert("El número secreto es mayor");
24         }
25     }
26     //Contador de intentos
27     //intentos = intentos + 1;
28     intentos++;
29     //palabraVeces = "veces";
30     if (intentos > maximosIntentos) {
31         alert(`Haz llegado al máximo de ${maximosIntentos} intentos`);
32         break;
33     }
34 }
35

```

- Observando el código original (propuesta por las clases de one) podemos observar que tenemos un par de sitios donde no tenemos variables y ahora cambiaremos esos sitios.
- Crearemos una variable para definir los límites entre x número, en este caso sera: numeroMaximoPosible = 10 (10 para mantener la funcionalidad del juego). Debido a esta creación de variable, iremos reemplazando el Math.ranbom()*10 por numeroMaximoPosible

```

js app.js > ...
1 //Variables
2 let numeroMaximoPosible = 10;
3 let numeroSecreto = Math.floor(Math.random()*numeroMaximoPosible)+1;
4 let numeroUsuario = 0;
5 let intentos = 1;
6 //let palabraVeces = "vez";
7 let maximosIntentos = 3;
8
9 console.log(numeroSecreto);
10
11 while(numeroUsuario != numeroSecreto){
12     numeroUsuario = parseInt(prompt(`Me indicas un número entre 1 y ${numeroMaximoPosible} por favor.`));
13
14     console.log(typeof(numeroUsuario));
15
16     if (numeroUsuario == numeroSecreto) {
17         //Acertamos, fue verdadera la condición
18         alert(`Acertaste, el número es: ${numeroUsuario}. Lo hiciste en ${intentos} ${intentos == 1 ? 'vez': "veces"}`);
19     } else {
20         // condición adicional para guiar la respuesta
21         if (numeroUsuario > numeroSecreto){
22             alert('El número secreto es menor');
23         } else {
24             alert("El número secreto es mayor");
25         }
26         //Contador de intentos
27         //intentos = intentos + 1;
28         intentos++;
29         //palabraVeces = "veces";
30         if (intentos > maximosIntentos) {
31             alert(`Haz llegado al máximo de ${maximosIntentos} intentos`);
32             break;
33         }
34     }
35 }
36

```

4. Lo que aprendimos

¡Ha llegado el momento de celebrar tu gran logro!

En este curso de lógica, has superado todas las barreras y te has sumergido en la programación. Has aprendido cómo configurar el entorno de desarrollo hasta la creación de un juego completo con diferentes interacciones.

Y lo mejor de todo es que ahora eres parte de la comunidad de desarrolladores que tienen acceso a un vasto ecosistema de herramientas, documentación y soporte técnico.

¡Guau, cuántas cosas geniales!

Aquí te dejo una frase para que celebremos esta victoria: "Si siempre intentas ser normal, nunca descubrirás lo extraordinario que puedes llegar a ser." (Maya Angelou)

5. Proyecto final

- Podemos descarga el proyecto final: <https://github.com/alura-es-cursos/2034-logica-programacion-1/tree/Aula5>

6. Conclusión

- Llevamos un simple juego de adivinanza numérica a un código ejecutable, donde aplicamos los conceptos y curiosidades en este código

