**Curso de Lógica de programación: sumérgete en la programación con JavaScript**

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**

1. Muestra una alerta con el mensaje "¡Bienvenida y bienvenido a nuestro sitio web!".
2. Declara una variable llamada nombre y asígnale el valor "Lua".
3. Crea una variable llamada edady asígnale el valor 25.
4. Establece una variable numeroDeVentas y asígnale el valor 50.
5. Establece una variable saldoDisponible y asígnale el valor 1000.
6. Muestra una alerta con el texto "¡Error! Completa todos los campos".
7. 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 .
8. Utiliza un prompt para preguntar el nombre del usuario y almacénalo en la variable nombre.
9. Pide al usuario que ingrese su edad usando un prompt y almacénala en la variable edad.
10. Ahora, si la edad es mayor o igual a 18, muestra una alerta con el mensaje "¡Puedes obtener tu licencia de conducir!".Si necesitas ayuda, las soluciones a las actividades estarán disponibles en la sección

**SOLUCION**

1. **Desafío: hora de practicar**
2. alert("¡Bienvenida y bienvenido a nuestro sitio web!");
3. *let* nombre = "lua";
4. *let* edad = 25;
5. *let* numeroDeVentas = 50;
6. *let* saldoDisponible = 1000;
7. alert("¡Error! Completa todos los campos");
8. *let* mensajeDeError = "¡Error! Completa todos los campos";
9. alert(mensajeDeError);
10. nombre = prompt("cual es tu nombre de usuario:");
11. edad = prompt("cual es tu edad?");
12. if (edad>=18) {
13. alert("Sr. "+nombre+" con edad "+edad+" ¡Puedes obtener tu licencia de conducir!");
14. }

La lectura de la documentación oficial de un lenguaje de programación es fundamental para que un desarrollador de software pueda utilizarlo de manera efectiva.

En el caso de JavaScript, esta documentación proporciona una amplia gama de información, incluyendo conceptos sobre sintaxis, las bibliotecas y las funcionalidades disponibles en el lenguaje.

Algunos enlaces útiles a la documentación oficial de JavaScript incluyen:

* [La documentación del Lenguaje JavaScript](https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/What_is_JavaScript)

<https://developer.mozilla.org/es/docs/Learn_web_development/Core/Scripting/What_is_JavaScript>

* [Guía de JavaScript: qué es y cómo aprender el lenguaje más popular del mundo.](https://www.aluracursos.com/blog/guia-de-javascript)

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

Al leer la documentación oficial, obtenemos una comprensión más profunda del lenguaje y sus características. Esto facilita la escritura de código más limpio, claro y seguro. La documentación también puede ayudar al desarrollador a ahorrar tiempo, evitando la necesidad de buscar en foros u otras fuentes de información no confiables.

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.

**02 condicionales y concatenación**

Archivos del aula 2:

<https://github.com/alura-es-cursos/2034-logica-programacion-1/tree/Aula2>

**Desafío: hora de practicar**

Programar requiere práctica. Hemos creado una lista adicional de actividades (no obligatorias) para que practiques y refuerces aún más tu aprendizaje.

¿Listo para practicar?

**Desafíos:**

1. Pregunta 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!".
2. Verifica si un número ingresado por el usuario es positivo o negativo. Muestra una alerta informativa.
3. Crea un sistema de puntuación para un juego. Si la puntuación es mayor o igual a 100, muestra "¡Felicidades, has ganado!". En caso contrario, muestra "Intentalo nuevamente para ganar.".
4. Crea un mensaje que informe al usuario sobre el saldo de su cuenta, utilizando un template string para incluir el valor del saldo.
5. Pide al usuario que ingrese su nombre mediante un prompt. Luego, muestra una alerta de bienvenida usando ese nombre.

Si necesitas ayuda, las opciones de solución de las actividades estarán disponibles en la sección “Opinión de la persona instructora”.

Solución:

<https://github.com/alura-es-cursos/desafios-logica-programacion/blob/main/desafio-2>

**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.

Info:

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

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.

En la próxima lección:

Aprenderemos cómo trabajar con estructuras de repetición.

**03 Loops y tentativas**

Link de archivos:

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

**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? Desafíos:

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.

Si necesitas ayuda, las opciones de solución para las actividades estarán disponibles en la sección "Opinión del instructor".

<https://github.com/alura-es-cursos/desafios-logica-programacion/blob/main/desafio-3>

**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 banama…

**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 |

¿Te gustó este conocimiento y quieres más? Rafa Ballerini tiene un artículo increíble hablando sobre ‘[Cómo usar operadores de comparación en Javascript](https://www.aluracursos.com/blog/como-utilizar-operadores-de-comparacion-en-javascript)’ que vale la pena leer

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

**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.

En la próxima lección:

Aprendamos cómo evitar código duplicado y aplicar buenas prácticas de programación. Además, ¡generemos un número verdaderamente "aleatorio"!

**04. Buenas prácticas en programación**

Link:

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

**Desafío: Hora de practicar**

**Link:** [**https://github.com/alura-es-cursos/desafios-logica-programacion/blob/main/desafio-4**](https://github.com/alura-es-cursos/desafios-logica-programacion/blob/main/desafio-4)

**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](https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/Math/random). 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.

Link: <https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/Math/random>

**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.

En la próxima lección:

Tendrás la oportunidad de poner en práctica todo el conocimiento que has adquirido resolviendo un desafío. ¡Hasta entonces, buena suerte!

**05. Desafio**

Link: <https://github.com/alura-es-cursos/2034-logica-programacion-1/tree/Aula5>

**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)**

Ahora, califica el curso, descarga tu certificado y celebra este logro en grande.

* [Christian Velasco](https://app.aluracursos.com/user/christian-velasco)
* [Leonardo Castillo](https://app.aluracursos.com/user/ljcl79)

**Para saber más: diferencias entre HTML, CSS y JS**

HTML, CSS y JavaScript (JS) son tres tecnologías fundamentales para el desarrollo web.

**HTML** (HyperText Markup Language) es **el lenguaje de marcado** utilizado para estructurar el contenido de una página web. Este lenguaje emplea etiquetas para definir la jerarquía de los elementos de una página. Originado en 1991 por Tim Berners-Lee, HTML es como el esqueleto de una página web: trabaja organizando el contenido, como títulos, párrafos, imágenes y enlaces, para que todo esté ordenado y en su lugar.

**CSS** (Cascading Style Sheets), creado en 1995, se destaca como **un lenguaje de estilos**. Separa la presentación de la estructura: es responsable de la presentación y **estilización** de los elementos, ya que permite controlar colores, fuentes, diseños y otros aspectos visuales. Es como vestir la estructura HTML con ropa elegante para que la página se vea exactamente como deseas.

**JavaScript**, por su parte, es el **único lenguaje de programación** de los tres. Es responsable de agregar **interactividad y dinamismo** a las páginas. Permite la creación de funcionalidades como animaciones, validaciones de formularios y actualizaciones de contenido en tiempo real.

El papel de cada una de estas tecnologías en la construcción de aplicaciones web, en resumen, es el siguiente: HTML estructura el contenido, CSS define el estilo y el diseño, y JavaScript agrega funcionalidades dinámicas.

| **Tecnología** | **Función** | **Ejemplos de Uso** |
| --- | --- | --- |
| HTML | Lenguaje de marcado para estructurar contenido. | Título, Párrafo |
| CSS | Lenguaje de estilos para presentación y estilización. | p { color: blue; } div { background: #f2f2f2; } |
| JavaScript | Lenguaje de programación para interactividad. | function iniciarJuego() { ... } |

Aprender estas tecnologías es crucial para cualquier persona interesada en el desarrollo web, ya que forman la base del entorno digital. Comprender HTML es necesario para estructurar y organizar el contenido de una página de manera semántica, lo que es vital para la accesibilidad, la indexación en motores de búsqueda y una experiencia consistente en diferentes dispositivos.

CSS permite a las personas desarrolladoras controlar el diseño y la apariencia de sus sitios web, lo que permite la creación de diseños atractivos y receptivos que se adaptan a diferentes tamaños de pantalla.

Por último, JavaScript nos capacita a crear experiencias interactivas y dinámicas, mejorando la usabilidad al ofrecer funcionalidades más avanzadas a los usuarios.

La competencia en HTML, CSS y JavaScript permite dar vida a ideas de manera creativa y funcional, lo que brinda a usuarios finales una experiencia atractiva e intuitiva.

Además, estas habilidades son altamente valoradas en el mercado laboral, ya que la demanda de desarrolladores web calificados es constante, dada la continua importancia de la presencia en línea para empresas y particulares. Por lo tanto, aprender y mejorar estas tecnologías es una inversión significativa para cualquiera que desee destacarse en el campo del desarrollo web.

Si deseas obtener más información sobre las características de estas tres tecnologías, Rafa Ballerini escribió [un artículo que enfatiza estas diferencias](https://www.aluracursos.com/blog/html-css-javascript-cuales-son-las-diferencias).

**Desafío: hora de practicar**

En una carrera de desarrollo de software, la práctica constante de la lógica de programación desempeña un papel fundamental en la construcción de bases sólidas.

La lógica de programación no solo te permite la creación de algoritmos eficientes y soluciones elegantes, sino que también desarrolla la capacidad de pensar de manera estructurada y analítica. Esta habilidad es esencial para enfrentar desafíos complejos y convertir problemas abstractos en implementaciones tangibles.

Con esto en mente, hemos creado una lista de actividades (no obligatorias) centradas en la práctica para mejorar aún más tu experiencia de aprendizaje. ¡Vamos a practicar!

**Desafíos**

1. Descarga otro proyecto haciendo clic en [este enlace](https://github.com/alura-es-cursos/js-curso-2/tree/main) y ábrelo en Visual Studio Code.
2. Cambia el contenido de la etiqueta h1 con document.querySelector y asigna el siguiente texto: "Hora del Desafío".
3. Crea una función que muestre en la consola el mensaje "El botón fue clicado" siempre que se presione el botón "Console".
4. Crea una función que se ejecute cuando se haga clic en el botón "prompt", preguntando el nombre de una ciudad de Brasil. Luego, muestra una alerta con el mensaje concatenando la respuesta con el texto: "Estuve en {ciudad} y me acordé de ti".
5. Crea una función que muestre una alerta con el mensaje: "Yo amo JS" siempre que se presione el botón "Alerta".
6. Al hacer clic en el botón "suma", pide 2 números y muestra el resultado de la suma en una alerta.

Si necesitas ayuda, las opciones de solución de las actividades estarán disponibles en la sección "Opinión del instructor".

Link: solución del desafio

<https://github.com/alura-es-cursos/js-curso/blob/main/1.md>

**Lo que aprendimos**

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

* Cargamos el proyecto con HTML y CSS para crear la estructura y el estilo de la página que usaremos durante el curso.
* Aprendimos a manipular el contenido de los elementos HTML, como H1 y párrafos (p), utilizando JavaScript para cambiar dinámicamente el texto mostrado en la página.
* Comprendimos la importancia de la interacción entre HTML y JavaScript para crear páginas web dinámicas e interactivas.

En la próxima aula:

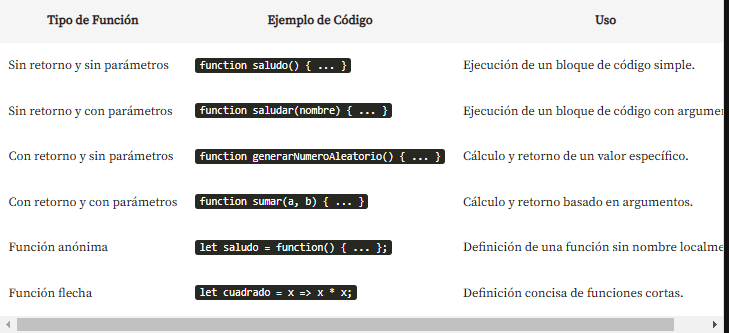
Aprenderemos qué son las funciones de manera práctica y cuál es su importancia en la carrera de desarrollo de software.

**02 funciones**

Link: <https://github.com/alura-es-cursos/2035-logica-programacion-2/tree/Aula2>

**Para saber más: funciones con JavaScript**

Las funciones juegan un papel central en la programación y en el desarrollo de software, permitiendo encapsular bloques de código reutilizables y ejecutables. Pueden ser definidas para realizar tareas específicas, desde cálculos simples hasta la manipulación compleja de datos. Las funciones en JavaScript pueden tener o no parámetros, así como devolver o no valores.



Su uso es esencial para crear aplicaciones dinámicas, interactivas y eficientes en JavaScript. Sin embargo, no es necesario que crees todos los tipos de funciones en este momento. Sigue las lecciones y ten en cuenta que estas variaciones existen, y a medida que surja la necesidad, profundizaremos cada vez más en la lógica y JavaScript.

**Desafío: hora de practicar**

Saber cómo usar funciones es uno de los conceptos fundamentales en la programación y el desarrollo de software. Las funciones desempeñan un papel crucial en la organización, modularización y reutilización del código.

Con eso en mente, hemos creado una lista de actividades (opcionales) centradas en la práctica para mejorar aún más tu experiencia de aprendizaje. ¡Así que vamos a practicar!

**Desafíos:**

1. Crear una función que muestre "¡Hola, mundo!" en la consola.
2. Crear una función que reciba un nombre como parámetro y muestre "¡Hola, [nombre]!" en la consola.
3. Crear una función que reciba un número como parámetro y devuelva el doble de ese número.
4. Crear una función que reciba tres números como parámetros y devuelva su promedio.
5. Crear una función que reciba dos números como parámetros y devuelva el mayor de ellos.
6. Crear una función que reciba un número como parámetro y devuelva el resultado de multiplicar ese número por sí mismo.

Si necesitas ayuda, las opciones de solución a las actividades estarán disponibles en la sección "Opinión del instructor".

Para ayudarte a revisar tus códigos, te dejo disponible una posible solución de los desafíos en este [enlace](https://github.com/alura-es-cursos/js-curso/blob/main/2.md) para que puedas construir o validar tus soluciones. ¡Buena suerte en tus estudios!

<https://github.com/alura-es-cursos/js-curso/blob/main/2.md>

<https://github.com/alura-es-cursos/js-curso/blob/main/2.md>

**Lo que aprendimos**

En esta aula:

* Creamos una función en JavaScript que recibe 2 parámetros para generar un número aleatorio y entendimos cómo utilizar la estructura de una función para encapsular un bloque de código reutilizable.
* Utilizamos la palabra clave "return" dentro de la función para devolver el valor del número aleatorio generado y creamos otra función llamada "exhibirMensageInicial" que no tiene ni retorno ni parámetros.
* Comprendimos la diferencia entre funciones con retorno y funciones sin retorno, y aprendimos cómo llamar e invocar la función para mostrar un mensaje inicial en la pantalla.
* Vimos cómo los parámetros y los valores de retorno son importantes para la comunicación entre funciones y el resto del programa.

En la próxima aula:

Aprenderemos cómo reiniciar el juego y reforzaremos todo el conocimiento que hemos adquirido hasta ahora.

**03 reiniciando el juego**

Link: <https://github.com/alura-es-cursos/2035-logica-programacion-2/tree/Aula3>

**Para saber más: template strings**

Los Template strings, o plantillas de cadenas, son características comunes en muchos lenguajes de programación y sistemas de plantillas. Estas plantillas ofrecen una manera eficiente de crear cadenas de texto donde es posible incluir marcadores o espacios reservados que posteriormente serán reemplazados por valores concretos. Estos marcadores, por lo general, se encuentran delimitados por caracteres especiales, como llaves, corchetes o signos de dólar, y desempeñan el papel de puntos de inserción para datos dinámicos.

En la gran mayoría de los lenguajes de programación, Template strings simplifican la tarea de concatenar valores variables en una cadena, sin requerir concatenación manual de cadenas o conversiones explícitas de tipos. Esto contribuye a que el código sea más legible y menos propenso a errores, lo cual es particularmente valioso en diversas situaciones, como la generación de mensajes de usuario, la construcción de consultas SQL dinámicas o la creación de documentos HTML.

Un ejemplo ilustrativo de esto se encuentra en JavaScript, donde puedes emplear las comillas invertidas (backticks) para crear plantillas de cadenas. Esto te permite incrustar expresiones dentro de la cadena utilizando el formato ${}. A continuación, un ejemplo práctico:

**const** nombre = "Juan";

**const** edad = 30;

**const** mensaje = `Hola, soy ${nombre} y tengo ${edad} años.`;

console.**log**(mensaje);

// Salida: Hola, soy Juan y tengo 30 años.

Copia el código

En este ejemplo, la plantilla de cadena permite insertar dinámicamente los valores de las variables nombre y edad en la cadena de texto.

Si quieres profundizar más sobre Template strings te dejamos este [artículo](https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Template_literals).

<https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Template_literals>

**09Desafío: hora de practicar**

Desafíos: Saber cómo usar funciones es uno de los conceptos fundamentales en la programación y el desarrollo de software. Las funciones desempeñan un papel crucial en la organización, modularización y reutilización del código.

Con eso en mente, hemos creado una lista de actividades (opcionales) centradas en la práctica para mejorar aún más tu experiencia de aprendizaje. ¡Así que vamos a practicar!

**Desafíos:**

1. Crea una función que calcule el índice de masa corporal (IMC) de una persona a partir de su altura en metros y peso en kilogramos, que se recibirán como parámetros.
2. Crea una función que calcule el valor del factorial de un número pasado como parámetro.
3. Crea una función que convierta un valor en dólares, pasado como parámetro, y devuelva el valor equivalente en reales(moneda brasileña,si deseas puedes hacerlo con el valor del dólar en tu país). Para esto, considera la cotización del dólar igual a R$4,80.
4. Crea una función que muestre en pantalla el área y el perímetro de una sala rectangular, utilizando la altura y la anchura que se proporcionarán como parámetros.
5. Crea una función que muestre en pantalla el área y el perímetro de una sala circular, utilizando su radio que se proporcionará como parámetro. Considera Pi = 3,14.
6. Crea una función que muestre en pantalla la tabla de multiplicar de un número dado como parámetro.

Si necesitas ayuda, las opciones de solución a las actividades estarán disponibles en la sección "Opinión del instructor".

Solución link:

<https://github.com/alura-es-cursos/js-curso/blob/main/3.md>

En esta aula:

* Aprendimos cómo crear un programa para verificar si el 'intento' ingresado es igual al número secreto definido.Utilizamos estructuras condicionales para tomar decisiones basadas en el resultado de esta comparación;
* Creamos una variable para almacenar la cantidad de intentos realizados por el usuario;
* Hemos visto la importancia de consultar la documentación del lenguaje y las bibliotecas utilizadas en el desarrollo del programa. La documentación es una fuente valiosa de información que nos ayuda a comprender conceptos y a utilizar correctamente las funcionalidades disponibles.

En la próxima aula:

Vamos a aprender cómo funcionan las listas (o arrays, en inglés) y cuán fundamentales son en la carrera de desarrollo de software.

**04.Listas**

Link:

<https://github.com/alura-es-cursos/2035-logica-programacion-2/tree/Aula4>

**07 Para saber más: ¿Vamos a leer más sobre listas?**

Un array, arreglo o lista en JavaScript es una estructura de datos que permite almacenar y organizar múltiples valores en una sola variable. Los valores en un array pueden ser de cualquier tipo de dato, como números, strings, objetos, otros arrays, y así sucesivamente. Los arrays en JavaScript son indexados, lo que significa que cada valor dentro de ellos está asociado con un índice numérico, generalmente comenzando desde el índice 0.

**Creando un array**

Puedes crear un array en JavaScript declarando una variable y asignándole valores entre corchetes [].

**let** frutas = ["Manzana", "Uva", "Naranja"];

Copia el código

Accediendo a los valores Los elementos de un array se acceden mediante índices numéricos, que comienzan en 0.

| **Índice** | **Elemento** |
| --- | --- |
| 0 | "Manzana" |
| 1 | "Uva" |
| 2 | "Naranja" |

console.**log**(frutas[0]); // Salida: "Manzana"

console.**log**(frutas[2]); // Salida: "Naranja"

Copia el código

**Añadiendo nuevos elementos**

Para agregar un elemento al final del array, puedes usar el método push.

frutas.**push**("Fresa");

console.**log**(frutas); // Salida: ["Manzana", "Uva", "Naranja", "Fresa"]

Copia el código

**Eliminando el último elemento**

Para eliminar el último elemento, puedes usar el método pop.

frutas.**pop**();

console.**log**(frutas); // Salida: ["Manzana", "Uva", "Naranja"]

Copia el código

**¿Qué lenguajes de programación utilizan arrays?**

Aquí tienes una lista de algunos lenguajes de programación que utilizan arrays:

* JavaScript
* Python
* Java
* C++
* C#
* Ruby
* PHP
* Swift
* Kotlin
* Go

Otros lenguajes también admiten el uso de arrays o estructuras de datos similares para almacenar colecciones de valores. Aprender sobre arrays es importante porque desempeñan un papel fundamental en el desarrollo de aplicaciones de software.

Las listas o arrays proporcionan una manera eficiente de almacenar y acceder a conjuntos de datos, lo que permite a los programadores organizar la información de manera lógica y manipular estos datos de manera efectiva.

Con el conocimiento de cómo trabajar con arrays, es posible crear algoritmos más poderosos, resolver problemas de programación de manera más eficiente y crear aplicaciones más dinámicas e interactivas.

**08 Desafío: hora de practicar**

Saber cómo usar las funciones es uno de los conceptos fundamentales en la programación y el desarrollo de software. Las funciones desempeñan un papel crucial en la organización, modularización y reutilización del código.

Con eso en mente, hemos creado una lista de desafíos (no obligatorios) centrados en la práctica para mejorar aún más tu experiencia de aprendizaje. ¡Vamos a practicar entonces!

**Desafíos**

1. Crea una lista vacía llamada "listaGenerica".
2. Crea una lista de lenguajes de programación llamada "lenguagesDeProgramacion con los siguientes elementos: 'JavaScript', 'C', 'C++', 'Kotlin' y 'Python'.
3. Agrega a la lista "lenguagesDeProgramacion los siguientes elementos: 'Java', 'Ruby' y 'GoLang'.
4. Crea una función que muestre en la consola todos los elementos de la lista "lenguagesDeProgramacion.
5. Crea una función que muestre en la consola todos los elementos de la lista "lenguagesDeProgramacion en orden inverso.
6. Crea una función que calcule el promedio de los elementos en una lista de números.
7. Crea una función que muestre en la consola el número más grande y el número más pequeño en una lista.
8. Crea una función que devuelva la suma de todos los elementos en una lista.
9. Crea una función que devuelva la posición en la lista donde se encuentra un elemento pasado como parámetro, o -1 si no existe en la lista.
10. Crea una función que reciba dos listas de números del mismo tamaño y devuelva una nueva lista con la suma de los elementos uno a uno.
11. Crea una función que reciba una lista de números y devuelva una nueva lista con el cuadrado de cada número.

Si necesitas ayuda, las soluciones a las actividades estarán disponibles en la sección "Opinión del instructor".

Para ayudarte a revisar tus códigos, te dejo disponible una posible solución de los desafíos en este [enlace](https://github.com/alura-es-cursos/js-curso/blob/main/4.md) para que puedas construir o validar tus soluciones.

<https://github.com/alura-es-cursos/js-curso/blob/main/4.md>

**09 Lo que aprendimos**

En esta aula:

* Aprendimos a trabajar con listas en JavaScript, incluyendo cómo agregar elementos, obtener el tamaño de la lista, acceder a elementos específicos y la importancia de conocer la posición del último elemento. También destacamos la importancia de leer la documentación y comprender los conceptos relacionados con el manejo de listas en JavaScript.
* La importancia de trabajar con funciones en la programación.
* Cómo generar un número secreto a través de una función.
* El uso del método push para almacenar el número generado al final de la lista.
* Cómo verificar si un número ya existe en la lista utilizando el método includes de JavaScript.
* La aplicación de la recursividad para generar un nuevo número válido.
* Como implementar un límite en el juego para evitar el bucle infinito.
* Como implementar una variable adicional para limitar la cantidad de veces que se puede jugar antes de reiniciar el juego.

En la próxima aula:

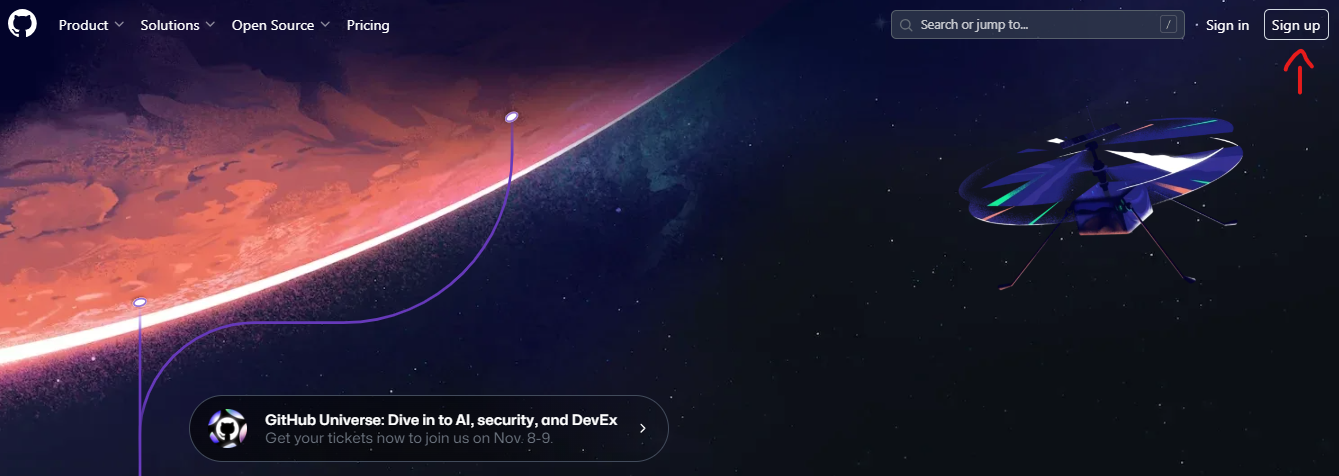
Tendrás la oportunidad de poner en práctica todo el conocimiento que has adquirido resolviendo un desafío. ¡Hasta entonces, buena suerte!.

**03Para saber más: creando una cuenta en GitHub y Vercel (paso a paso)**

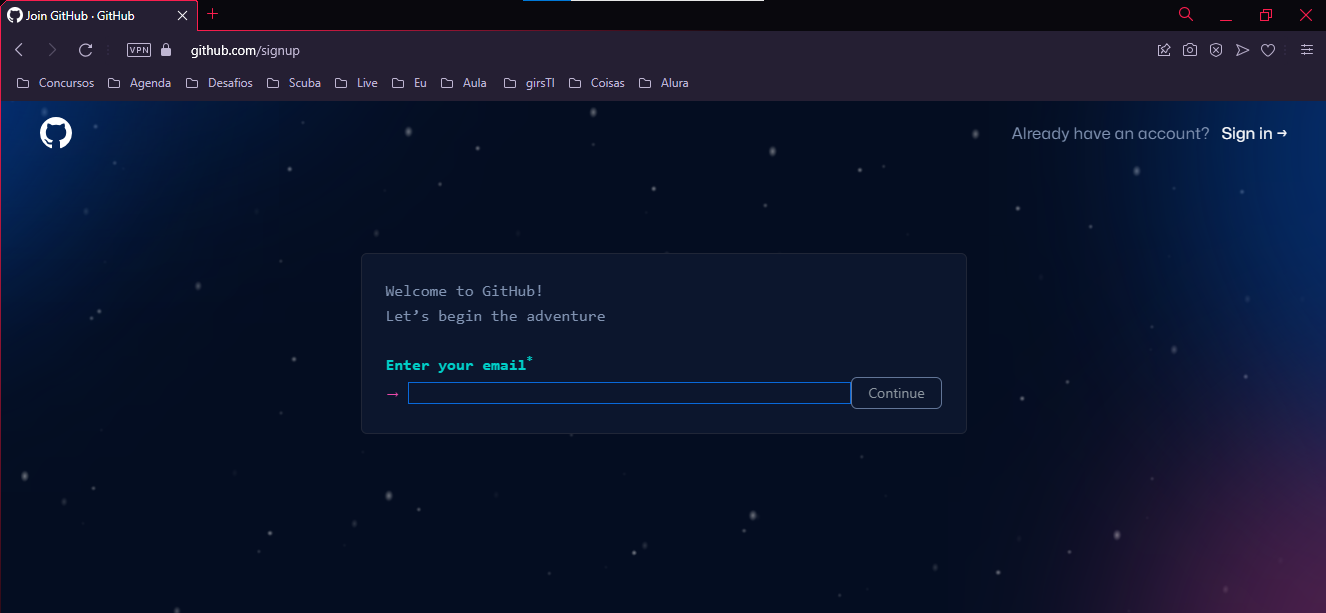
GitHub es una plataforma ampliamente utilizada para alojar y colaborar en proyectos de código fuente por personas desarrolladoras de software. Ofrece funciones de control de versiones que permiten a equipos de programadores trabajar de manera eficiente, realizando cambios, revisiones y resolviendo conflictos de código de forma colaborativa. ¡Es como la 'red social' de los desarrolladores!

Además, GitHub facilita el seguimiento de problemas y solicitudes de extracción, lo que hace que el proceso de desarrollo sea más transparente y organizado.

El primer paso para crear una cuenta en GitHub es acceder al sitio web de GitHub a través de este [enlace](https://github.com/) y hacer clic en 'Sign up'.



A continuación, verás una página como esta:



Haz lo siguiente:

* Ingresa tu correo electrónico.
* Crea una contraseña.
* Ingresa un nombre de usuario.
* Si deseas recibir actualizaciones y anuncios de productos por correo electrónico, escribe 's' para sí o 'n' para no.
* Haz clic en el botón Continuar.

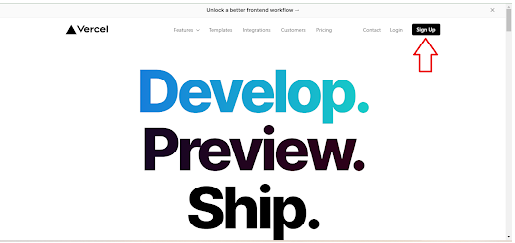
Después de completar tu información personal y hacer clic en Continuar, haz clic en Create Account y recibirás una confirmación de registro por correo electrónico. Después de validar tu cuenta, podrás utilizar los servicios de GitHub :)

**Creando una cuenta en Vercel**

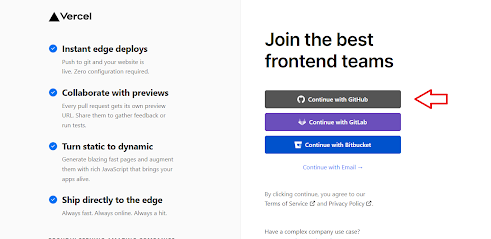
Por su parte, Vercel es una plataforma de alojamiento e implementación de aplicaciones web modernas, especializada en ofrecer experiencias de carga rápida y rendimiento optimizado. A menudo se utiliza para alojar sitios web estáticos y aplicaciones de una sola página (SPA), aprovechando técnicas avanzadas y eficientes para los usuarios finales.

Vercel simplifica la implementación continua y la escalabilidad automática, permitiendo que los desarrolladores se centren en la creación de excelentes experiencias digitales sin preocuparse por la complejidad subyacente de la infraestructura.

El primer paso para crear una cuenta en Vercel es acceder al sitio web de Vercel a través de este enlace y hacer clic en 'Sign up'.



Después de hacer clic en 'Sing up', verás una página como esta:



Luego, haz clic en 'Continue with GitHub' y listo, ¡tu cuenta ha sido creada! Ahora solo tienes que utilizar los servicios de Vercel :)l.

Link proyecto final completo:

<https://github.com/alura-es-cursos/2035-logica-programacion-2/tree/Aula5>

**¡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!

"La valentía no es la ausencia del miedo; es la persistencia a pesar del miedo…" (Desconocido)

Ahora, califica el curso, descarga tu certificado y celebra este logro en grande

* [Christian Velasco](https://app.aluracursos.com/user/christian-velasco)
* [Leonardo Castillo](https://app.aluracursos.com/user/ljcl79)