# MANUAL DEL LENGUAJE ELM
# (PixelArcade)

# por
# ROBOTICSO . CO

| | |
|---|---|
| Dueño del documento: | ROBOTICSO .CO |

| | |
|---|---|
| ROBOTICSO . CO | Integrantes:<br>- Becerra Sipiran, Cledy Elizabeth<br>- Camero Gonzalez, Ian Gabriel<br>- Oviedo Sivincha, Massiel<br>- Ramos Villena, Sergio Leandro<br>- Villanueva Borda, Harold Alejandro |

Versión

| VERSIÓN | FECHA | DESCRIPCIÓN | CONTROLADOR |
|---|---|---|---|
| 1.0 | 11-07-2023 | Guia del Lenguaje Implementado | Yessenia Deysi Yari Ramos |

# ÍNDICE

# 1. INTRODUCCIÓN A ELM

Elm es un lenguaje de programación funcional y declarativo diseñado para construir aplicaciones web robustas y confiables. Con su enfoque en la arquitectura de modelo de vista actualizable (Model-View-Update), Elm proporciona una forma elegante y segura de crear interfaces de usuario interactivas.

A diferencia de otros lenguajes de programación, Elm se centra en eliminar errores comunes en el desarrollo de aplicaciones web al brindar un sistema de tipos fuertes y estáticos. Esto significa que muchas de las fallas y excepciones que pueden ocurrir en otros lenguajes se capturan durante la compilación, lo que permite detectar y corregir problemas antes de que se ejecuten.

Elm también se destaca por su capacidad para gestionar eficientemente el estado de la aplicación. Utiliza un modelo de arquitectura basado en mensajes, donde los cambios en el estado de la aplicación se realizan mediante acciones claras y predecibles. Esta estructura facilita el razonamiento sobre el código y permite una fácil escalabilidad a medida que las aplicaciones crecen en complejidad.

Además de su enfoque en la confiabilidad y la facilidad de mantenimiento, Elm ofrece una amplia gama de bibliotecas y herramientas que simplifican el desarrollo web. Con Elm, los desarrolladores pueden crear interfaces de usuario interactivas y responsivas sin tener que preocuparse por problemas comunes, como el manejo manual de eventos y la manipulación directa del DOM.

En resumen, Elm es un lenguaje de programación que combina la elegancia de la programación funcional con la seguridad de los tipos estáticos. Su enfoque en la confiabilidad y la escalabilidad lo convierte en una excelente opción para construir aplicaciones web modernas y robustas.

# 2. INSTALACIÓN

## 2.1. Localmente

Para instalar Elm en tu sistema, puedes seguir estos pasos:

1. Primero, asegúrate de tener instalado Node.js en tu computadora. Puedes descargar Node.js desde su sitio web oficial (https://nodejs.org) e instalarlo siguiendo las instrucciones para tu sistema operativo.
2. Una vez que tienes Node.js instalado, abre una terminal o línea de comandos en tu computadora.
3. Ejecuta el siguiente comando para instalar Elm a través del administrador de paquetes de Node.js (npm):


Este comando instalará Elm globalmente en tu sistema, lo que te permitirá usarlo en cualquier proyecto.

1. Después de que la instalación haya finalizado, puedes verificar que Elm se haya instalado correctamente ejecutando el siguiente comando:

```
C:\Users\massi>elm --version
0.19.1
```

Este comando debería mostrar la versión de Elm instalada en tu sistema.
¡Y eso es todo! Ahora tienes Elm instalado y listo para usar en tu computadora. Puedes comenzar a crear proyectos de Elm y utilizar las herramientas y bibliotecas disponibles en el ecosistema de Elm para el desarrollo de aplicaciones web.

### 2.2. Online

Existen entornos de desarrollo en línea para Elm que son útiles para aquellos que desean probar rápidamente el código Elm sin la necesidad de instalar herramientas localmente. Permiten comenzar rápidamente, experimentar con el lenguaje y compartir proyectos con otros desarrolladores de manera fácil y rápida. También son útiles para realizar demostraciones o pruebas rápidas sin la necesidad de configuraciones complicadas.Entre ellas, tenemos:

1. **Ellie App (https://ellie-app.com/)**
   Ellie es una herramienta en línea que permite escribir y ejecutar código Elm de forma rápida y sencilla. Algunas ventajas de Ellie incluyen:
   a. Interfaz sencilla y fácil de usar.
   b. Permite compartir y colaborar en proyectos Elm con otras personas a través de enlaces.
   c. Proporciona una visualización en tiempo real del resultado de la compilación y la ejecución del código.

2. **CodeSandbox (https://codesandbox.io/)**
   CodeSandbox es una plataforma en línea que admite múltiples lenguajes de programación, incluido Elm. Sus ventajas incluyen:
   a. Un entorno de desarrollo completo y basado en la web.
   b. Permite escribir y ejecutar código Elm de manera rápida y eficiente.
   c. Proporciona funcionalidades adicionales, como la gestión de dependencias y la posibilidad de trabajar en proyectos más complejos con múltiples archivos.

3. **Replit (https://replit.com/)**
   Replit es una plataforma en línea para desarrollar y ejecutar código en varios lenguajes, incluido Elm. Sus ventajas incluyen:
   a. Un entorno de desarrollo en línea con todas las herramientas necesarias para escribir, compilar y ejecutar código Elm.
   b. Permite colaborar con otros desarrolladores en tiempo real en proyectos Elm compartidos.
   c. Ofrece opciones de almacenamiento y gestión de proyectos.

## 3. TIPOS DE DATOS PRIMITIVOS

Estos son los tipos de datos primitivos básicos en Elm. Además de estos, Elm también proporciona estructuras de datos más complejas y tipos personalizados que se pueden definir en el lenguaje. Estos tipos primitivos proporcionan los bloques de construcción fundamentales para modelar y manipular datos en Elm.

### 3.1.Simples

1. Números enteros (Int)
   Representa valores numéricos enteros sin decimales. Por ejemplo, 0, 10, -5.
2. Números de punto flotante (Float)
   Representa valores numéricos con decimales. Por ejemplo, 3.14, -2.5, 1.0.
3. Caracteres (Char)
   Representa un único carácter Unicode. Por ejemplo, 'a', 'B', '1'.
4. Cadenas de texto (String)
   Representa una secuencia de caracteres Unicode. Por ejemplo, "Hola", "Elm", "123".
5. Booleanos (Bool)
   Representa un valor de verdad, que puede ser verdadero (True) o falso (False). Estos son utilizados para expresar condiciones o lógica booleana.

```elm
-- Números enteros
myInt : Int
myInt = 42

-- Números de punto flotante
myFloat : Float
myFloat = 3.14

-- Caracteres
myChar : Char
myChar = 'A'

-- Cadenas de texto
myString : String
myString = "Hola, Elm!"

-- Booleanos
myBool : Bool
myBool = True

```

### 3.2. Compuestos

1. Listas (List)
   Representa una secuencia ordenada de valores del mismo tipo. Por ejemplo, [1, 2, 3], ['a', 'b', 'c'].
2. Tuplas (Tuple)
   Representa una colección ordenada de valores de diferentes tipos. Por ejemplo, (1, "Hola"), ('a', True, 3.14).

```
26  -- Listas
27  myList : List Int
28  myList = [1, 2, 3, 4, 5]
29
30  -- Tuplas
31  myTuple : (Int, String, Bool)
32  myTuple = (42, "Elm", True)
33
```

## 4.CONDICIONALES

Estas son las principales estructuras condicionales en Elm. Se utilizan para tomar decisiones basadas en condiciones y ejecutar diferentes ramas de código en función de esas decisiones.

### 4.1.Condicional (if…then…else)

Esta es la estructura condicional básica en Elm. Permite evaluar una condición y ejecutar una rama de código si la condición es verdadera (then) o ejecutar otra rama de código si la condición es falsa (else).

src › Main.elm

```
1  module Main exposing (..)
2
3  import Html exposing (text)
4
5
6  -- Función para determinar si un número es par o impar
7  checkEvenOdd : Int -> String
8  checkEvenOdd number =
9    if number % 2 == 0 then
10     "El número es par"
11   else
12     "El número es impar"
13
14
15 -- Función principal para mostrar el resultado
16 main =
17   text (checkEvenOdd 5)
```

### 4.2.Condicional (case…of)

Esta estructura condicional se utiliza para hacer coincidir un valor con diferentes patrones y ejecutar un código correspondiente en función del patrón que coincide.

src > Main.elm

```elm
module Main exposing (..)

import Html exposing (text)

-- Función para determinar el día de la semana según el número
getWeekday : Int -> String
getWeekday day =
  case day of
    1 ->
      "Lunes"
    2 ->
      "Martes"
    3 ->
      "Miércoles"
    4 ->
      "Jueves"
    5 ->
      "Viernes"
    6 ->
      "Sábado"
    7 ->
      "Domingo"
    _ ->
      "Día no válido"

-- Función principal para mostrar el resultado
main =
  text (getWeekday 3)
```

### 4.2.Condicional (let…in…)

Aunque no es estrictamente una estructura condicional, se puede utilizar para asignar un valor condicionalmente y luego utilizar ese valor en una expresión.

src > Main.elm

```elm
module Main exposing (..)

import Html exposing (text)

-- Función para calcular el área de un rectángulo
calculateArea : Float -> Float -> Float
calculateArea width height =
  let
    area = width * height
  in
    area

-- Función principal para mostrar el resultado
main =
  text (toString (calculateArea 5.0 3.0))
```

## 5. BUCLES

En Elm, no existen bucles tradicionales como for o while como los que se encuentran en otros lenguajes de programación imperativos. En su lugar, Elm promueve el enfoque funcional y utiliza recursión y funciones de orden superior para realizar operaciones repetitivas.

Elm ofrece varias funciones de orden superior que permiten trabajar con listas de manera recursiva, como **List.map, List.filter, List.foldl y List.foldr**. Estas funciones permiten aplicar transformaciones o realizar operaciones en cada elemento de una lista.

Además, Elm proporciona la función **List.range** que genera una lista de números en un rango específico. Esto puede ser utilizado en combinación con funciones de orden superior para realizar tareas repetitivas.
Aquí hay un ejemplo que muestra cómo utilizar recursión y funciones de orden superior para realizar una operación repetitiva en una lista:

src › Main.elm

```elm
module Main exposing (..)

import Html exposing (text)

-- Función para sumar todos los números en una lista
sumList : List Int -> Int
sumList numbers =
  case numbers of
    [] ->
      0
    x :: xs ->
      x + sumList xs

-- Función principal para mostrar el resultado
main =
  let
    numbers = [1, 2, 3, 4, 5]
  in
    text (toString (sumList numbers))

```

En este ejemplo, se define la función sumList que toma una lista de números enteros y devuelve la suma de todos los números en la lista. Utiliza recursión para recorrer la lista y sumar los elementos.

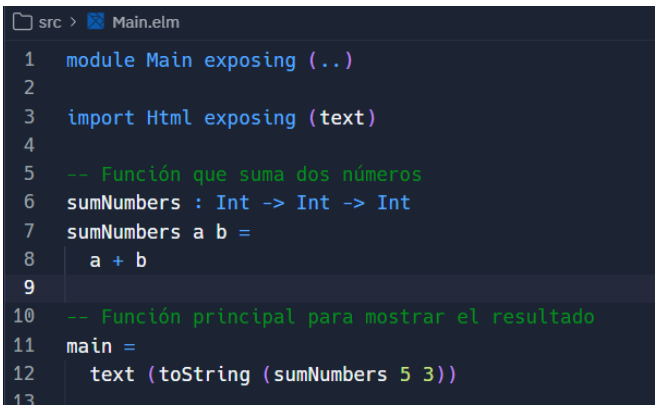
En la función principal main, se crea una lista de números [1, 2, 3, 4, 5] y se llama a la función sumList para obtener la suma de los números. El resultado se muestra en el navegador utilizando la función text del módulo Html.

Si ejecutas este código en un proyecto Elm y lo visualizas en el navegador, verás el resultado de la suma de los números: 15.

Recuerda que en Elm se fomenta el enfoque funcional y el uso de recursión y funciones de orden superior en lugar de bucles tradicionales. Esto ayuda a crear código más claro, conciso y más fácil de razonar.

## 6. FUNCIONES

En Elm, puedes definir funciones utilizando la sintaxis **nombreFuncion argumentos = expresion**. Aquí hay un ejemplo de cómo hacerlo:




En este ejemplo, se define la función sumNumbers que toma dos argumentos a y b, ambos de tipo Int, y devuelve la suma de esos dos números. La expresión a + b realiza la suma.

En la función principal main, se llama a la función sumNumbers con los argumentos 5 y 3, y el resultado se muestra en el navegador utilizando la función text del módulo Html.

Puedes definir funciones con cualquier número de argumentos y tipos de datos en Elm. Además, Elm también admite funciones anónimas o lambdas utilizando la sintaxis **\argumentos -> expresion.**

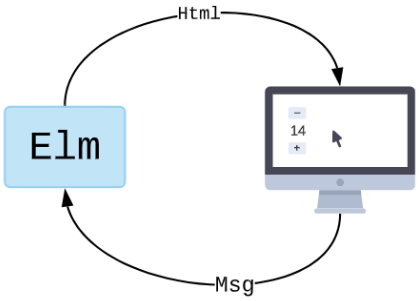
Recuerda que en Elm, las funciones son inmutables y no tienen efectos secundarios. Cada vez que se llama a una función con los mismos argumentos, siempre producirá el mismo resultado, lo que facilita el razonamiento y la depuración del código.

## 7. ARQUITECTURA ELM

Elm Architecture es un patrón para la arquitectura de programas interactivos, como aplicaciones web y juegos.
Esta arquitectura parece surgir naturalmente en Elm. En lugar de que alguien lo inventara, los primeros programadores de Elm siguieron descubriendo los mismos patrones básicos en su código. ¡Fue un poco espeluznante ver a las personas terminar con un código bien diseñado sin planificar con anticipación!
Entonces The Elm Architecture es fácil en Elm, pero es útil en cualquier proyecto front-end. De hecho, proyectos como Redux se han inspirado en The Elm Architecture, por lo que es posible que ya hayas visto derivados de este patrón. El punto es que, incluso si finalmente no puede usar Elm en el trabajo todavía, obtendrá mucho de usar Elm e internalizar este patrón.

### 7.1.Patrón Basico

Los programas de Elm siempre se ven así:




El programa Elm produce HTML para mostrar en la pantalla, y luego la computadora envía mensajes de lo que está sucediendo. "¡Hacen clic en un botón!"

Sin embargo, ¿qué sucede dentro del programa Elm? Siempre se divide en tres partes:

- Modelo: el estado de su aplicación
- Ver: una forma de convertir su estado en HTML
- Actualizar: una forma de actualizar su estado en función de los mensajes

Estos tres conceptos son el núcleo de The Elm Architecture.
Los siguientes ejemplos mostrarán cómo usar este patrón para la entrada del usuario, como botones y campos de texto. ¡Hará esto mucho más concreto!

## 8.ANEXOS

8.1 Slides Manual