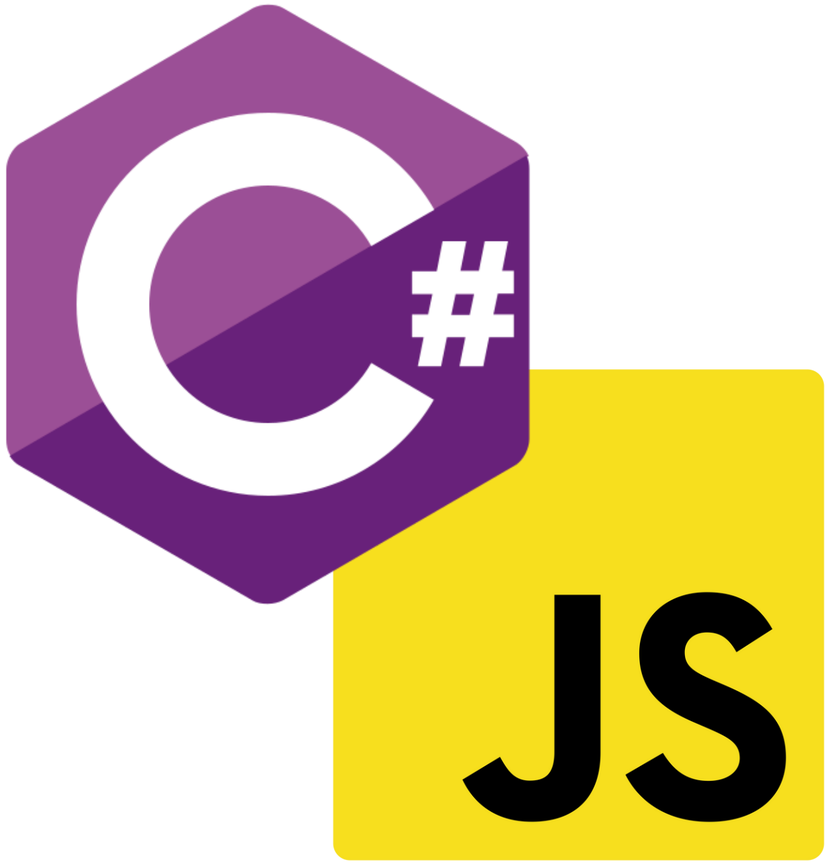
Sintaxis y Semántica de los Lenguajes

C# Y JAVASCRIPT

Presentado por Esteban Gomez Lousa, Micaela Agüero, Tomas Fiestas y Gianfranco Riccelli

2023

Docente: Ing. Roxana Leituz



Objetivos

- Dar una introducción histórica sobre los lenguajes elegidos
- Hacer un análisis comparativo del algoritmo de ordenamiento de vectores y arboles binarios
- Dar una recomendación

Breve historia JavaScript

- Creado en 1995 por Brendan Eich
- Lenguaje de scripting para navegadores web
- Estandarizacion bajo el nombre ECMAScript
- No tiene relacion directa con Java

Netscape

Breve historia C#

- Creado por Anders Hejlsberg y su equipo en la década de 1990
- Facilidad de uso, la seguridad y la capacidad de aprovechar al máximo la plataforma .NET
- Introducción de LINQ (Language Integrated Query)
- Patrones de diseño asincrónicos y await/async

COMPARACION GENERAL

	
Compilado	Interpretado
Estatico	Dinamico
OOP	Scripting/OOP
Detecta errores en cualquier momento	Detecta errores despues de la ejecucion

Ordenamiento de vectores

```javascript
function bubbleSort(arr) {
    const n = arr.length;
    for (let i = 0; i < n - 1; i++) {
        for (let j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                const temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}

function printVector(arr) {
    console.log(arr.join(' '));
}

const vector = [10, 5, 7, 9, 4];

console.log("Vector original:");
printVector(vector);

console.time("Tiempo de ejecución");

const memoryBefore = process.memoryUsage().heapUsed;

bubbleSort(vector);

const memoryAfter = process.memoryUsage().heapUsed;

console.timeEnd("Tiempo de ejecución");

console.log("\nVector ordenado:");
printVector(vector);

const memoryUsed = memoryAfter - memoryBefore;
console.log(`\nMemoria utilizada (bytes): ${memoryUsed}`);
```

```
10 5 7 9 4
Tiempo de ejecución: 0.322ms

Vector ordenado:
4 5 7 9 10

Memoria utilizada (bytes): 1776
```

Ordenamiento de vectores

```
class BubbleSortExample
{
    static void Main(string[] args)
    {
        int[] vector = { 10, 5, 7, 9, 4 };

        Console.WriteLine("Vector original:");
        PrintVector(vector);

        long memoryBefore = GC.GetTotalMemory(true);

        BubbleSort(vector);

        long memoryAfter = GC.GetTotalMemory(true);

        Console.WriteLine("\nVector ordenado:");
        PrintVector(vector);

        long memoryUsed = memoryAfter - memoryBefore;
        Console.WriteLine($"\nMemoria utilizada (bytes): {memoryUsed}");
    }

    static void BubbleSort(int[] arr)
    {
        int n = arr.Length;
        for (int i = 0; i < n - 1; i++)
        {
            for (int j = 0; j < n - i - 1; j++)
            {
                if (arr[j] > arr[j + 1])
                {
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
            }
        }
    }
```

```
    static void PrintVector(int[] arr)
    {
        foreach (int num in arr)
        {
            Console.Write(num + " ");
        }
        Console.WriteLine();
    }
}
```

Post Ejecución

Vector original:
10 5 7 9 4

Vector ordenado:
4 5 7 9 10

Memoria utilizada (bytes): -104

Y con un tiempo de ejecución de 0.006 segundos.

Ordenamiento de vectores

Lenguaje	Tiempo de ejecución	Memoria Utilizada
Javascript	0.332ms	1776
C#	6ms	104

ARBOL BINARIO

```
void Main(){}
class TreeNode
{
    public int Value;
    public TreeNode Left;
    public TreeNode Right;

    public TreeNode(int value)
    {
        Value = value;
        Left = null;
        Right = null;
    }
}

class BinarySearchTree
{
    public TreeNode Root;

    public BinarySearchTree()
    {
        Root = null;
    }

    public void Insert(int value)
    {
        TreeNode newNode = new TreeNode(value);

        if (Root == null)
        {
            Root = newNode;
            return;
        }
```

```
        TreeNode current = Root;
        while (true)
        {
            if (value < current.Value)
            {
                if (current.Left == null)
                {
                    current.Left = newNode;
                    return;
                }
                current = current.Left;
            }
            else
            {
                if (current.Right == null)
                {
                    current.Right = newNode;
                    return;
                }
                current = current.Right;
            }
        }
    }
}
```

```
class Program
{
    static void Main(string[] args)
    {
        int[] values = { 1, 8, 6, 34, 12, 21, 72, 15, 50, 45 };
        BinarySearchTree tree = new BinarySearchTree();

        foreach (int value in values)
        {
            tree.Insert(value);
        }

        PrintTree(tree.Root, 0);
    }

    static void PrintTree(TreeNode node, int indent)
    {
        if (node == null)
        {
            return;
        }

        PrintTree(node.Right, indent + 4);
        Console.WriteLine(new string(' ', indent) + node.Value);
        PrintTree(node.Left, indent + 4);
    }
}
}
```

```
5 Tiempo de ejecución: 34 ms
6 Uso máximo de memoria: 26452 KB
```

ARBOL BINARIO

```
Tiempo de ejecuci▯n: 13 ms
Uso aproximado de memoria: 44 KB
```

JS

ARBOL BINARIO

Lenguaje	Tiempo de ejecución	Memoria Utilizada
Javascript	13ms	44 KB
C#	34ms	26452 KB

Cuestionario

- Que lenguaje es compilado y cual interpretado?
- En que ámbitos es mejor usar C#?
- En que ámbitos es mejor usar JavaScript?
- Que lenguaje detecta errores incluso antes de la ejecución?