# **Informe sobre Scala y JavaScript: Historia, Sintaxis, y Funciones de Ordenamiento**

## **1. Introducción**

Este informe presenta una visión comparativa entre los lenguajes de programación Scala y JavaScript, enfocándose en su historia, sintaxis y la manera en que manejan las funciones de ordenamiento. Además, se exploran los algoritmos de ordenamiento utilizados por estos lenguajes y se comparan sus rendimientos.

## **2. Breve Historia de los Lenguajes**

### **Scala**

* **Lanzamiento:** 2004
* **Creador:** Martin Odersky
* **Objetivo:** Combinar los paradigmas de programación orientada a objetos y funcional.
* **Nombre:** Abreviación de “Scalable Language”, reflejando su flexibilidad y escalabilidad.
* **Características Clave:**
  + **Programación Mixta:** Combina paradigmas funcionales y orientados a objetos, con una sintaxis poderosa y flexible.
  + **Compatibilidad con JVM:** Se ejecuta en la Java Virtual Machine (JVM), integrándose con el ecosistema Java.
  + **Compilación a JavaScript:** Escalable a aplicaciones web y dispositivos con soporte de navegador.
  + **Tipos Estáticos:** Sistema de tipos estáticos que asegura errores en tiempo de compilación.

### **JavaScript**

* **Lanzamiento:** 1995
* **Creador:** Brendan Eich en Netscape Communications Corporation
* **Nombre Original:** LiveScript (renombrado a JavaScript para asociarlo con la popularidad de Java).
* **Objetivo:** Añadir interactividad a las páginas web.
* **Características Clave:**
  + **Lenguaje Dinámico:** Ligero y dinámico, ideal para contenido interactivo en la web.
  + **Ejecución en Navegadores y Servidores:** Utilizado tanto en el navegador como en el servidor (Node.js).
  + **Estándares ECMAScript:** Sigue especificaciones como ES5 y ES6 (ES2015), con mejoras continuas.
  + **Frameworks y Bibliotecas:** Base para numerosos frameworks como React, Angular y Vue.js.

## **3. Funciones de Ordenamiento en Scala y JavaScript**

### **Timsort**

* **Descripción:** Algoritmo híbrido que combina **Merge Sort** e **Insertion Sort**.
* **Funcionamiento:**
  + **División en "Runs":** La lista se divide en subsecuencias pequeñas ordenadas (runs).
  + **Ordenación de Runs Pequeñas:** Utiliza **Insertion Sort** para ordenar estas pequeñas runs.
  + **Combinación de Runs Ordenadas:** Usa **Merge Sort** para fusionar las runs ordenadas en una lista completa.

### **Ordenamiento en Scala**

* **Método:** sorted()
* **Algoritmo para Tipos Simples:** **Dual-Pivot Quicksort**
  + **Características:** Complejidad similar a Quick Sort, pero con menor número de intercambios de elementos.
* **Algoritmo para Tipos Complejos:** **Timsort**
  + **Uso:** Para colecciones de tipos complejos como clases.
  + **Personalización:** Puede recibir un parámetro de tipo Ordering para definir el criterio de ordenación.

### **Ordenamiento en JavaScript**

* **Método:** sort()
* **Algoritmo:** **Timsort** en motores modernos como V8 (Google Chrome y Node.js)
* **Características:**
  + **Complejidad Temporal Promedio:** O(n log n)
  + **Manejo de Datos Parcialmente Ordenados:** Eficiente en listas con patrones repetitivos.
  + **Función de Comparación:** Necesaria para asegurar el orden correcto en tipos complejos y números, ya que el método sort() por defecto convierte los elementos a cadenas y los ordena alfabéticamente.

## **4. Comparación de Rendimiento**

### **JavaScript**

* **Método:** sort()
* **Rendimiento:** Timsort es eficiente para datos con segmentos ordenados o patrones repetitivos. Puede requerir ajustes (como una función de comparación) para datos numéricos o tipos complejos.

### **Scala**

* **Método:** sorted()
* **Rendimiento:**
  + **Dual-Pivot Quicksort:** Eficiente para tipos primitivos con una buena tasa de intercambios.
  + **Timsort:** Ofrece una mejor performance para colecciones con tipos complejos.

**Ejemplo Comparativo de Rendimiento:**

**AGREGAR DATA**

## **5. Conclusiones**

* **Scala y JavaScript** tienen metodologías distintas para el ordenamiento de datos.
* **Timsort** es común en ambos lenguajes para colecciones más complejas.
* **Dual-Pivot Quicksort** en Scala es eficiente para tipos primitivos.
* El rendimiento puede variar según el tipo de datos y el algoritmo de ordenamiento utilizado.

## **6. Bibliografía**

* **SCALA**
  + **BNF:** [**https://www.scala-lang.org/files/archive/spec/2.11/13-syntax-summary.html**](https://www.scala-lang.org/files/archive/spec/2.11/13-syntax-summary.html) **o** [**https://github.com/scala/scala/blob/2.11.x/spec/13-syntax-summary.md**](https://github.com/scala/scala/blob/2.11.x/spec/13-syntax-summary.md)
  + **SORT:** [**https://www.scala-lang.org/api/current/scala/collection/SeqView$$Sorted.html**](https://www.scala-lang.org/api/current/scala/collection/SeqView$$Sorted.html)
  + **Historia/características:**
    - [**https://docs.scala-lang.org/es/tour/tour-of-scala.html**](https://docs.scala-lang.org/es/tour/tour-of-scala.html)
    - [**https://es.m.wikipedia.org/wiki/Scala\_(lenguaje\_de\_programaci%C3%B3n)**](https://es.m.wikipedia.org/wiki/Scala_(lenguaje_de_programaci%C3%B3n))
    - [**https://www.javatpoint.com/history-of-scala**](https://www.javatpoint.com/history-of-scala)
    - **https://docs.scala-lang.org/scala3/book/scala-for-javascript-devs.html**
* **Información Javascript**
  + **Historia:**
    - [**https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First\_steps/What\_is\_JavaScript**](https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript)
    - [**https://dev.to/dboatengx/history-of-javascript-how-it-all-began-92a**](https://dev.to/dboatengx/history-of-javascript-how-it-all-began-92a)
    - **https://en.wikipedia.org/wiki/JavaScript**
  + **BNF:** [**https://262.ecma-international.org/#sec-grammar-summary**](https://262.ecma-international.org/#sec-grammar-summary) **o** [**https://ecma-international.org/wp-content/uploads/ECMA-262\_15th\_edition\_june\_2024.pdf#[{%22num%22%3A2268%2C%22gen%22%3A0}%2C{%22name%22%3A%22XYZ%22}%2C0%2C747%2C0**](https://ecma-international.org/wp-content/uploads/ECMA-262_15th_edition_june_2024.pdf#%5B%7B%22num%22%3A2268%2C%22gen%22%3A0%7D%2C%7B%22name%22%3A%22XYZ%22%7D%2C0%2C747%2C0)**]**
  + **SORT:** [**https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\_Objects/Array/sort**](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/sort)
  + [**https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\_Objects/Array/toSorted**](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/toSorted)
  + [**https://desarrolloweb.com/articulos/ordenacion-arrays-javascript-sort**](https://desarrolloweb.com/articulos/ordenacion-arrays-javascript-sort)