# A Brief Survey of Modern Lanaguages 2024

Satya Komatineni, 9/2/2024

Take a brief look at the modern languages, such as Rust, Mojo, Elixir, Carbon.

## Table of Contents



## Introduction

As time passes, inexorably new paradigms emerge. Intelligence doesn't stay stale. It moves in time and place.

Computer languages have always been the hot bed of new thought.

I wanted to take stock of where languages are, recently (2024).

Few questions I want to explore are:

1. Why are there so many new languages lately? Or are there that many?
2. If there are, what are they?
3. General purpose languages vs specialized languages?
4. Front end vs backend languages?
5. What are the key features and advantages of the newer specialized languages?
6. If I were to retool, how should I consider these new languages?
7. What are some programming languages to watch out for?
8. What is the likely hood of their future adoption?
9. Whcih ones should you care about?
10. How simple or hard is the learning curve for these languages?
11. Am I missing something if I am not aware of the advancement in these newer languages?

# A list of newer languages: in 2024

| Language | Popularity | Main Goal | By Who? | Age | Learning Curve |
|---|---|---|---|---|---|

| Language | Popularity | Main Goal | By Who? | Age | Learning Curve |
|---|---|---|---|---|---|
| **Mojo** | Low (Emerging) | High-performance computing, AI/ML workloads, Python compatibility. | Modular | 1 year (2023) | Moderate |
| **Carbon** | Low (Emerging) | Successor to C++ with modern features and safety. | Google | 2 years (2022) | High |
| **Zig** | Low (Growing) | Systems programming with a focus on safety, simplicity, and optimal performance. | Andrew Kelley | 8 years (2016) | Moderate to High |
| **Crystal** | Moderate | Combines Ruby-like syntax with the performance of a compiled language. | Ary Borenszweig, .. | 10 years (2014) | Moderate |
| **Swift** | High | iOS and macOS application development, offering modern syntax and performance. | Apple | 10 years (2014) | Low to Moderate |
| **Julia** | Moderate | High-performance numerical and scientific computing. | An MIT Team | 11 years (2012) | Moderate |
| **TypeScript** | Very High | Type-safe JavaScript for large-scale applications. Supports gradual typing for JavaScript. | Microsoft | 11 years (2012) | Low to Moderate |
| **Elixir** | Moderate | Scalable and maintainable applications, particularly in concurrent environments. | José Valim | 12 years (2012) | Moderate |
| **Dart** | Moderate | Mobile and web app development (especially with Flutter). | Google | 12 years (2011) | Low to Moderate |
| **Rust** | High | Systems programming with safety and concurrency. Ideal for performance-critical tasks. | Mozilla | 13 years (2010) | High |
| **Kotlin** | High | Modern development for Android and JVM applications, with interoperability with Java. | JetBrains | 13 years (2011) | Low to Moderate |
| **Go** | High | Simplified systems programming, cloud services, and concurrency. | Google | 14 years (2009) | Low |

| Language | Popularity | Main Goal | By Who? | Age | Learning Curve |
|---|---|---|---|---|---|
| **F#** | Moderate | Functional-first language with .NET compatibility for data processing and analytical tasks. | MS Research | 18 years (2005) | Moderate |
| **Scala** | Moderate | A language combining object-oriented and functional programming on the JVM. | Martin Odersky | 20 years (2003) | High |
| **C#** | High | General-purpose programming, Windows applications, enterprise software. | Microsoft | 22 years (2002) | Moderate |
| **C++** | Very High | Systems programming, performance-critical applications, and large-scale software development. | Bjarne Stroustrup | 38 years (1985) | High |

😃 Here is a comma separated list of those languages, in case you want some furthe research

```
Mojo, Carbon, Zig, Crystal, Swift, Julia, TypeScript, Elixir, Dart, Rust, Kotlin,
Go, F#, Scala, C#, C++
```

# Languages in Game development

| Language | Popularity | Who is Using It? | Why |
|---|---|---|---|
| **C++** | Very High | Epic Games (Unreal Engine), Blizzard Entertainment, Valve, Electronic Arts (Frostbite Engine) | High-performance, low-level control, vast ecosystem of libraries, widely used for AAA games and game engines. |
| **C#** | Very High | Unity Technologies (Unity Engine), Microsoft (Minecraft), CD Projekt Red (Gwent) | Strongly supported by Unity, cross-platform capabilities, easier learning curve than C++, large community support. |
| **JavaScript** | High | Web-based game developers, Kongregate, Poki, Facebook Instant Games | Easily accessible, widely supported in browsers, large ecosystem of libraries and frameworks like Phaser.js. |
| **TypeScript** | High | Microsoft (PlayFab), browser-based game developers, Roblox (partially) | Provides type safety on top of JavaScript, improving development experience and scalability for larger projects. |

| Language | Popularity | Who is Using It? | Why |
|---|---|---|---|
| **Rust** | High | Embark Studios, Ready At Dawn, game engine developers exploring new technologies | Memory safety, high performance without a garbage collector, increasing interest in building safe and efficient game engines. |
| **Swift** | Moderate | Apple Arcade developers, mobile game developers focusing on iOS | Optimized for iOS development, modern language features, strong performance on Apple hardware. |
| **Go** | Moderate | Game backend services (e.g., games with heavy networking or server requirements), MMO game servers | Efficient concurrency model, suitable for scalable server-side applications, simple syntax and performance for backend services. |
| **Kotlin** | Moderate | Android game developers, smaller indie studios targeting Android platforms | Modern language features, seamless Java interoperability, optimized for Android development. |
| **Mojo** | Low (Emerging) | Emerging interest from game developers focusing on AI/ML integrations and high-performance computing | High-performance computing, emerging language tailored for AI/ML workloads, Python compatibility makes it approachable. |

# Languages for Web servers

| Language | Popularity | Who is Using It? | Why |
|---|---|---|---|
| **JavaScript (Node.js)** | Very High | Netflix, LinkedIn, PayPal, Walmart, Trello | Non-blocking, event-driven architecture allows handling a large number of concurrent connections efficiently. Widely used for building fast, scalable web servers and APIs. |
| **Python** | Very High | Instagram, Spotify, YouTube, Dropbox | Popular for web development with frameworks like Django and Flask; easy to use, with a large ecosystem of libraries for rapid web application development. |
| **Java** | Very High | Twitter, LinkedIn, Amazon, eBay | Strong concurrency support and a mature ecosystem make it suitable for building scalable, high-performance web servers and enterprise applications. |
| **Go** | High | Google, Dropbox, Uber, Twitch, Cloudflare | Efficient concurrency model (goroutines) and simple syntax make it ideal for building scalable, high-performance web servers and microservices. |

| Language | Popularity | Who is Using It? | Why |
|---|---|---|---|
| **C# (.NET)** | High | Microsoft, Stack Overflow, Siemens | Well-supported for building scalable web servers, especially on Windows platforms, with strong support for concurrency and asynchronous programming. |
| **TypeScript** | High | Slack, Microsoft, Asana | Adds type safety to JavaScript, improving development experience and robustness for large-scale web server projects, especially with frameworks like Node.js. |
| **PHP** | High | Facebook, WordPress, Wikipedia, Mailchimp | Designed for server-side scripting; widely used in web development for building dynamic websites and content management systems (CMS) like WordPress. |
| **Ruby** | Moderate | GitHub, Shopify, Airbnb, Hulu | Used in web development with Ruby on Rails; known for its ease of use and rapid development capabilities, particularly for startups and MVPs. |
| **Elixir** | Moderate | Discord, Bleacher Report, PagerDuty | Built on the Erlang VM (BEAM), designed for highly concurrent, fault-tolerant, real-time web applications and chat systems. |
| **Kotlin** | Moderate | Pinterest, Trello, Coursera | Modern features and JVM compatibility make it suitable for web servers, especially in environments where Java is traditionally used. |
| **Rust** | Moderate | Mozilla, Dropbox, Amazon, Discord | Memory safety and concurrency features make it suitable for building reliable, high-performance web servers. |
| **Scala** | Moderate | Twitter, LinkedIn, The Guardian | Combines functional and object-oriented programming, runs on the JVM, and is suitable for building concurrent and distributed web applications. |
| **Swift** | Low to Moderate | Apple, IBM, Lyft | Primarily used for iOS development, but also supports server-side applications with frameworks like Vapor, particularly on macOS or Linux servers. |
| **Haskell** | Low to Moderate | Standard Chartered Bank, Facebook, Barclays | Purely functional language with strong concurrency support; used in scenarios requiring high assurance, correctness, and mathematical rigor. |

| Language | Popularity | Who is Using It? | Why |
|---|---|---|---|
| **Perl** | Low | Craigslist, BBC, DuckDuckGo | Traditionally used for server-side scripting and web development, but less common in new web server projects today. |

# Lanauages for Correctness and Concurrency

| Language | Popularity | Who is Using It? | Why |
|---|---|---|---|
| **Go** | High | Google, Dropbox, Uber, Cloudflare | Go's concurrency model with goroutines and channels makes it suitable for parallelism. The language's simplicity and strict typing reduce bugs, and its built-in tools promote error-free, robust programming. |
| **Rust** | High | Mozilla, Dropbox, Amazon, Discord | Rust ensures memory safety and thread safety without a garbage collector. Its ownership model prevents data races and other concurrency errors, making it ideal for systems requiring reliability and fault tolerance. |
| **Elixir** | Moderate | Discord, Bleacher Report, PagerDuty | Built on the Erlang VM (BEAM), Elixir provides robust fault tolerance and concurrency support. Its functional programming model and immutable data structures reduce errors, making it suitable for real-time, distributed systems. |
| **Haskell** | Low to Moderate | Standard Chartered Bank, Facebook, Barclays | Haskell's strong type system and purely functional programming paradigm enforce correctness and reduce runtime errors. Its concurrency support is well-suited for parallel systems that require high assurance and reliability. |
| **Erlang** | Low to Moderate | WhatsApp, Ericsson, Cisco, Heroku | Designed for building fault-tolerant, distributed systems, Erlang's actor model provides massive concurrency. Its focus on reliability and process isolation makes it ideal for systems that must remain operational despite failures. |
| **Scala** | Low to Moderate | Twitter, LinkedIn, The Guardian | Scala's combination of functional and object-oriented programming allows for robust error handling and type safety. It runs on the JVM, providing access to strong concurrency libraries and frameworks like Akka, which support fault tolerance and distributed computing. |
| **F#** | Low to Moderate | Microsoft, Jet.com, Citigroup | F# is a functional-first language that runs on the .NET platform. It provides strong type safety and immutability, which help prevent runtime errors. It also supports parallel and asynchronous programming, making it suitable for fault-tolerant systems. |

# If I were a young programmer today (Still specialized languages, except TypeScript)...

| Name | Why Focus | Learning Curve |
|---|---|---|
| **Rust** | Rust ensures memory safety and concurrency without a garbage collector. Increasingly used for systems programming, infrastructure, and performance-critical applications. High demand for developers due to its safety features and growing adoption. | High |
| **TypeScript** | TypeScript adds type safety to JavaScript, making it easier to catch errors early and maintain large codebases. Widely used in web development, especially in enterprise-level applications. Modern web development often relies on TypeScript for robust, scalable projects. | Low to Moderate |
| **Go (Golang)** | Go is known for its simplicity, efficient concurrency model, and high performance. Ideal for building scalable web servers, cloud-native applications, and backend services. Strong demand in cloud computing and DevOps roles. | Low |
| **Kotlin** | Kotlin offers a modern syntax and is fully interoperable with Java, making it the preferred language for Android development. It's also expanding into multiplatform development, allowing for shared code across mobile, web, and server applications. | Low to Moderate |
| **Swift** | Swift is optimized for the Apple ecosystem and is essential for iOS, macOS, watchOS, and tvOS development. Its modern syntax and safety features reduce common programming errors. Growing community and continuous support from Apple. | Low to Moderate |
| **Elixir** | Elixir is built on the Erlang VM, providing robust concurrency and fault tolerance. It's well-suited for real-time applications, such as chat systems and messaging platforms. Increasing interest in industries requiring low-latency and highly reliable systems. | Moderate |
| **Dart** | Dart is the language behind Flutter, a popular framework for cross-platform mobile, web, and desktop applications. Learning Dart opens opportunities in mobile development, especially with Flutter’s growing adoption. Strong tooling and Google support. | Low to Moderate |
| **Julia** | Julia is designed for high-performance numerical and scientific computing, making it an excellent choice for data science, machine learning, and technical computing. Easy to learn for those familiar with Python, MATLAB, or R. Growing adoption in data-intensive fields. | Moderate |
| **Mojo** | Mojo is an emerging language focused on high-performance computing and AI/ML workloads while being Python-compatible. It's gaining attention in AI and data-intensive applications. Combining Python’s ease of use with lower-level language performance. | Moderate |

# Languages with explicit memory management, or No GC

These are

```
Rust
C++
C
```

No garbage collector (GC) systems are of interest for performance, predictability, and when control over resources are critical.

1. Performance and predictability
2. Explicit Memory management
3. Concurrency and multi-threading
4. Bringing it all together, a systems programming case

# My brief Take

Few languages on my radar, some immediately, and perhaps some soon.

## Python

Significant inroads in

1. Data and analysis (via Pandas, Numpy, Graphics libs)
2. AI (Pytorch, GenAI, RAG)
3. Becoming a main lingua franca

## TypeScript

This is now universal. Especially in front end applications (angular, react, etc.), or as part of the node.js server ecosystem.

## Dart

Due to Google's flutter as a mobile strong platform.

## Rust

It is really hard to write large systems and frameworks in a "fault tolerant" way. Rust seem to uniquely focusing on it with compile time checking along with additional features. This may be a compelling reason to look at it for large fault tolerant systems.

I suspect, however putting such explicit control on memory management, won't it create issues for average programmers?

## ELixir

I am curious, how and when concurrency can be a deciding factor? A bit cautious of its alternate programming model of Erlang.

## Mojo

1. It is python compatible
2. Apparently applicable in AI applications

## PowerShell

1. I love this language.
2. Hoping to use it to automate windows OS and cloud

# References

The markdown for this in github is here