ChatGPT

Programming Language Lineage Dataset

Executive summary

This dataset models “what was implemented in what,” across languages, compilers, runtimes, and major toolchains, with time ranges and source-backed evidence for each edge. It emphasizes well-documented bootstrap transitions like Go’s C-to-Go toolchain migration in Go 1.5, which removed almost all C from the compiler and runtime, apart from cgo and testing. [1] It also captures modern staged bootstraps, like Rust’s stage0-to-stage1-to-stage2 build, where the current compiler is built using a prebuilt “stage0” compiler. [2]

Where multiple implementations exist, the dataset includes parallel implementation records with separate edges and confidence scores. Where primary sources do not clearly pin down a historical detail, the dataset marks the link as “unspecified” in notes and assigns confidence below 0.8, rather than guessing. The “evidence_source” fields inside the dataset include official documentation, authoritative repos, and primary papers such as Stroustrup’s HOPL-II history of C++ and Thompson’s “Trusting Trust” lecture. [3]

JSON dataset


```
{
  "languages": [
    {
      "id": "lang:machine_code",
      "name": "Machine code",
      "first_release_year": 0,
      "current_primary_implementation_language": "unspecified",
      "notes": "Conceptual root node representing opaque binary seeds and CPU-executed code. Year is not meaningful."
    },
    {
      "id": "lang:assembly",
      "name": "Assembly language",
      "first_release_year": 0,
      "current_primary_implementation_language": "unspecified",
      "notes": "Conceptual root node for low-level human-readable code used in runtimes/compilers and bootstrap steps. Year is not meaningful."
    },
    {
      "id": "lang:bcpl",
      "name": "BCPL",
      "first_release_year": 1966,
      "current_primary_implementation_language": "unspecified",
```

```
      "notes": "Included as an upstream systems-language ancestor for B and C.
First-release year varies by historical source; treat as approximate."
    },
    {
      "id": "lang:b",
      "name": "B",
      "first_release_year": 1969,
      "current_primary_implementation_language": "unspecified",
      "notes":
"Included as a close ancestor to C in Unix lineage. First-release year varies by
historical source; treat as approximate."
    },
    {
      "id": "lang:c",
      "name": "C",
      "first_release_year": 1972,
      "current_primary_implementation_language": "C and C++ (major compilers/
toolchains)",
      "notes": "Early C compilers were bootstrapped through earlier system
languages (notably B) and then self-hosted on C on Unix; see Ritchie's history."
    },
    {
      "id": "lang:cxx",
      "name": "C++",
      "first_release_year": 1985,
      "current_primary_implementation_language": "C++ (self-hosted compilers)
plus C/C++ runtimes",
      "notes": "Cfront era relied on translation to C; later industrial
compilers are self-hosted or largely implemented in C++."
    },
    {
      "id": "tool:gcc",
      "name": "GCC",
      "first_release_year": 1987,
      "current_primary_implementation_language": "C++ and C (plus some other
languages in-tree)",
      "notes": "Node type: compiler suite/toolchain. GCC's repo language mix
shows substantial C++ and C today."
    },
    {
      "id": "tool:llvm",
      "name": "LLVM",
      "first_release_year": 2003,
      "current_primary_implementation_language": "C++",
      "notes": "Node type: compiler infrastructure. LLVM documentation and repo
show C++ as the core implementation language."
    },
    {
```

```
      "id": "tool:clang",
      "name": "Clang",
      "first_release_year": 2007,
      "current_primary_implementation_language": "C++",
      "notes": "Node type: C-family compiler frontend; launched as a new LLVM C-family frontend in 2007."
    },
    {
      "id": "lang:go",
      "name": "Go",
      "first_release_year": 2012,
      "current_primary_implementation_language": "Go and Assembly (toolchain/runtime)",
      "notes": "Go 1 released in 2012; Go 1.5 completed the C-to-Go toolchain translation and removed most C from compiler/runtime."
    },
    {
      "id": "lang:rust",
      "name": "Rust",
      "first_release_year": 2015,
      "current_primary_implementation_language": "Rust (compiler), with generated artifacts and build tooling",
      "notes": "Rust 1.0 released 2015. Early compiler work included OCaml and C; modern rustc is largely Rust and uses staged bootstrapping."
    },
    {
      "id": "tool:mrustc",
      "name": "mrustc",
      "first_release_year": 2016,
      "current_primary_implementation_language": "C++",
      "notes": "Node type: alternative Rust compiler designed for bootstrapping a recent rustc."
    },
    {
      "id": "lang:python",
      "name": "Python",
      "first_release_year": 1991,
      "current_primary_implementation_language": "C (CPython reference implementation)",
      "notes": "First public releases date to 1991; CPython remains the primary reference implementation."
    },
    {
      "id": "lang:ruby",
      "name": "Ruby",
      "first_release_year": 1995,
      "current_primary_implementation_language": "C and Ruby (CRuby/MRI; includes bytecode components)",
```

```
      "notes": "Ruby's canonical implementation is CRuby/MRI. Ruby 1.9
introduced the YARV-based VM as the official interpreter."
    },
    {
      "id": "lang:ocaml",
      "name": "OCaml",
      "first_release_year": 1996,
      "current_primary_implementation_language": "OCaml and C (compiler/
runtime), with some Assembly",
      "notes": "OCaml's core system includes compilers plus a C runtime; the
official repo shows OCaml + C as dominant."
    },
    {
      "id": "lang:haskell",
      "name": "Haskell",
      "first_release_year": 1990,
      "current_primary_implementation_language": "Multiple; GHC is dominant",
      "notes": "Haskell language first-release year is commonly cited as 1990
(Haskell 1.0). Treat as approximate due to variation across historical
summaries."
    },
    {
      "id": "tool:ghc",
      "name": "GHC",
      "first_release_year": 1992,
      "current_primary_implementation_language": "Haskell and C (runtime
system)",
      "notes": "Node type: compiler. GHC is widely cited as implemented largely
in Haskell with a C runtime system; primary confirmation varies by component."
    },
    {
      "id": "lang:java",
      "name": "Java",
      "first_release_year": 1996,
      "current_primary_implementation_language":
"Java (javac) plus C++ (HotSpot JVM)",
      "notes":
"Java 1.0 public release dated Jan 23, 1996 in Sun/JavaSoft press materials.
OpenJDK javac is in Java; HotSpot is in C++."
    },
    {
      "id": "tool:hotspot",
      "name": "HotSpot JVM",
      "first_release_year": 1999,
      "current_primary_implementation_language": "C++",
      "notes": "Node type: runtime/VM. OpenJDK sources state HotSpot code is
written in C++."
    },
```

```
    {
      "id": "lang:javascript",
      "name": "JavaScript",
      "first_release_year": 1995,
      "current_primary_implementation_language": "Multiple engines; major
engines are primarily C++ (with some Rust in SpiderMonkey)",
      "notes": "Year is widely cited as 1995 for initial creation; treat as
approximate in the absence of a single primary artifact in this dataset."
    },
    {
      "id": "tool:v8",
      "name": "V8",
      "first_release_year": 2008,
      "current_primary_implementation_language": "C++",
      "notes":
"Node type: JavaScript/WebAssembly engine. Official docs describe V8 as written
in C++."
    },
    {
      "id": "tool:spidermonkey",
      "name": "SpiderMonkey",
      "first_release_year": 1995,
      "current_primary_implementation_language": "C++ and Rust (plus some
JavaScript)",
      "notes": "Node type: JavaScript/WebAssembly engine. Project docs describe
an implementation spanning C++, Rust, and JavaScript."
    },
    {
      "id": "tool:javascriptcore",
      "name": "JavaScriptCore",
      "first_release_year": 2001,
      "current_primary_implementation_language": "C++ (with some C, Assembly,
Objective-C integration)",
      "notes": "Node type: JavaScript engine (WebKit). WebKit docs describe the
codebase as mostly C++ with bits of C and assembly, primarily in
JavaScriptCore."
    },
    {
      "id": "lang:csharp",
      "name": "C#",
      "first_release_year": 2002,
      "current_primary_implementation_language": "C# (Roslyn compiler) plus C/C+
+ (runtimes)",
      "notes": "Microsoft’s C# version history lists C# 1.0 released January
2002."
    },
    {
      "id": "tool:dotnet_runtime",
```

```
      "name": ".NET Runtime",
      "first_release_year": 2002,
      "current_primary_implementation_language": "C# with significant C++ and C
and some Assembly",
      "notes": "Node type: runtime/standard libraries/tooling. The dotnet/
runtime repo language mix shows C# dominant with major C++ and C portions."
    },
    {
      "id": "tool:roslyn",
      "name": "Roslyn",
      "first_release_year": 2014,
      "current_primary_implementation_language": "C# and Visual Basic .NET",
      "notes":
"Node type: C# and VB compiler platform. Repo language stats show primarily C#
and VB.NET."
    },
    {
      "id": "lang:kotlin",
      "name": "Kotlin",
      "first_release_year": 2016,
      "current_primary_implementation_language": "Kotlin (compiler), with Java
and native components",
      "notes": "Kotlin 1.0 released Feb 2016. The K2 compiler frontend is a
rewrite from scratch (per JetBrains)."
    },
    {
      "id": "lang:swift",
      "name": "Swift",
      "first_release_year": 2014,
      "current_primary_implementation_language": "C++ and Swift (compiler repo
mix)",
      "notes": "Swift was introduced in 2014 and open-sourced Dec 3, 2015. The
open-source swiftlang/swift repo shows a near-even C++ and Swift split."
    }
  ],
  "implementations": [
    {
      "id": "impl:c_unixcc_early",
      "target_language": "lang:c",
      "implementation_name": "Early Unix C compiler (bootstrapped through B
lineage)",
      "implementation_language": "B (and possibly earlier tools); unspecified
details vary",
      "time_period": "1972-1973",
      "status": "historical",
      "evidence_source": "https://www.bell-labs.com/usr/dmr/www/chist.html"
    },
    {
```

```
      "id": "impl:cxx_cfront",
      "target_language": "lang:cxx",
      "implementation_name": "cfront (C++-to-C translator)",
      "implementation_language": "C++ (front-end), output C",
      "time_period": "1982-1990s",
      "status": "historical",
      "evidence_source": "https://www.stroustrup.com/hopl2.pdf"
    },
    {
      "id": "impl:gcc_suite",
      "target_language": "tool:gcc",
      "implementation_name": "GNU Compiler Collection (suite)",
      "implementation_language": "C++ and C (plus others in-tree)",
      "time_period": "1987-present",
      "status": "active",
      "evidence_source":
"https://www.fsf.org/blogs/community/why-gcc-became-gcc | https://github.com/
gcc-mirror/gcc"
    },
    {
      "id": "impl:llvm_project",
      "target_language": "tool:llvm",
      "implementation_name": "LLVM project (core infrastructure)",
      "implementation_language": "C++",
      "time_period": "2003-present",
      "status": "active",
      "evidence_source":
"https://releases.llvm.org/1.0/docs/ReleaseNotes.html | https://llvm.org/docs/
GettingStarted.html | https://github.com/llvm/llvm-project"
    },
    {
      "id": "impl:clang_frontend",
      "target_language": "tool:clang",
      "implementation_name": "Clang (LLVM C-family frontend)",
      "implementation_language": "C++",
      "time_period": "2007-present",
      "status": "active",
      "evidence_source": "https://discourse.llvm.org/t/new-llvm-c-front-end-
clang/55248 | https://github.com/llvm/llvm-project"
    },
    {
      "id": "impl:go_gc_pre15",
      "target_language": "lang:go",
      "implementation_name": "gc toolchain (pre-Go 1.5 era)",
      "implementation_language": "C",
      "time_period": "2009-2014",
      "status": "historical",
      "evidence_source": "https://go.dev/doc/faq | https://go.dev/doc/go1.5"
```

```
    },
    {
      "id": "impl:go_gc_15plus",
      "target_language": "lang:go",
      "implementation_name": "gc toolchain (Go 1.5+)",
      "implementation_language": "Go and Assembly",
      "time_period": "2015-present",
      "status": "active",
      "evidence_source": "https://go.dev/doc/go1.5 | https://go.dev/blog/go1.5"
    },
    {
      "id": "impl:go_release_go1",
      "target_language": "lang:go",
      "implementation_name": "Go 1 release milestone",
      "implementation_language": "C (toolchain at that time)",
      "time_period": "2012-2015",
      "status": "historical",
      "evidence_source": "https://go.dev/blog/go1"
    },
    {
      "id": "impl:rustc_prehistory",
      "target_language": "lang:rust",
      "implementation_name": "Rust compiler (prehistory snapshot)",
      "implementation_language": "C and OCaml (plus smaller C++/Rust portions)",
      "time_period": "2006-2009",
      "status": "historical",
      "evidence_source": "https://github.com/graydon/rust-prehistory"
    },
    {
      "id": "impl:rustc_modern",
      "target_language": "lang:rust",
      "implementation_name": "rustc (modern)",
      "implementation_language": "Rust (with build scripts and small C
components)",
      "time_period": "2015-present",
      "status": "active",
      "evidence_source": "https://github.com/rust-lang/rust | https://blog.rust-
lang.org/2015/05/15/Rust-1.0/"
    },
    {
      "id": "impl:rust_bootstrap_stages",
      "target_language": "lang:rust",
      "implementation_name": "rustc staged bootstrapping (stage0->stage1-
>stage2)",
      "implementation_language": "Rust (built using a prebuilt stage0 compiler
binary)",
      "time_period": "2015-present",
      "status": "active",
```

```
      "evidence_source": "https://rustc-dev-guide.rust-lang.org/building/
bootstrapping/what-bootstrapping-does.html"
    },
    {
      "id": "impl:mrustc_bootstrap",
      "target_language": "tool:mrustc",
      "implementation_name": "mrustc alternative compiler",
      "implementation_language": "C++",
      "time_period": "2016-present",
      "status": "active",
      "evidence_source": "https://github.com/thepowersgang/mrustc"
    },
    {
      "id": "impl:cpython",
      "target_language": "lang:python",
      "implementation_name": "CPython reference interpreter",
      "implementation_language": "C and Python",
      "time_period": "1991-present",
      "status": "active",
      "evidence_source": "https://github.com/python/cpython | https://
en.wikipedia.org/wiki/History_of_Python"
    },
    {
      "id": "impl:cruby_mri",
      "target_language": "lang:ruby",
      "implementation_name": "CRuby/MRI (canonical Ruby implementation)",
      "implementation_language": "Ruby and C (plus smaller Rust/C++ portions in-
tree)",
      "time_period": "1995-present",
      "status": "active",
      "evidence_source": "https://github.com/ruby/ruby"
    },
    {
      "id": "impl:ruby_yarv",
      "target_language": "lang:ruby",
      "implementation_name": "YARV becomes official Ruby VM (Ruby 1.9 line)",
      "implementation_language": "C (VM core) and Ruby",
      "time_period": "2007-present",
      "status": "active",
      "evidence_source": "https://www.ruby-lang.org/en/news/2007/12/25/
ruby-1-9-0-released/ | https://en.wikipedia.org/wiki/YARV"
    },
    {
      "id": "impl:ocaml_core",
      "target_language": "lang:ocaml",
      "implementation_name": "OCaml compiler toolchain and runtime (official)",
      "implementation_language": "OCaml and C (plus some Assembly)",
      "time_period": "1996-present",
```

```
      "status": "active",
      "evidence_source": "https://github.com/ocaml/ocaml"
    },
    {
      "id": "impl:ghc",
      "target_language": "lang:haskell",
      "implementation_name": "GHC (dominant Haskell compiler)",
      "implementation_language": "Haskell (compiler) and C (runtime)",
      "time_period": "1992-present",
      "status": "active",
      "evidence_source":
"https://en.wikipedia.org/wiki/Glasgow_Haskell_Compiler | https://
www.haskell.org/ghc/"
    },
    {
      "id": "impl:openjdk_hotspot",
      "target_language": "tool:hotspot",
      "implementation_name": "HotSpot JVM (OpenJDK)",
      "implementation_language": "C++",
      "time_period": "2006-present",
      "status": "active",
      "evidence_source": "https://openjdk.org/jeps/8283227 | https://
openjdk.org/groups/hotspot/"
    },
    {
      "id": "impl:openjdk_javac",
      "target_language": "lang:java",
      "implementation_name": "javac (OpenJDK compiler)",
      "implementation_language": "Java",
      "time_period": "2006-present",
      "status": "active",
      "evidence_source": "https://github.com/openjdk/jdk/blob/master/src/
jdk.compiler/share/classes/com/sun/tools/javac/Main.java | https://openjdk.org/
groups/compiler/"
    },
    {
      "id": "impl:java_1_0_release",
      "target_language": "lang:java",
      "implementation_name": "Java 1.0 public release (JavaSoft)",
      "implementation_language": "unspecified (includes compiler and VM
binaries)",
      "time_period": "1996",
      "status": "historical",
      "evidence_source": "https://javaalmanac.io/jdk/1.0/pressrelease.pdf"
    },
    {
      "id": "impl:v8_engine",
      "target_language": "lang:javascript",
```

```
      "implementation_name": "V8 JavaScript/WebAssembly engine",
      "implementation_language": "C++",
      "time_period": "2008-present",
      "status": "active",
      "evidence_source": "https://v8.dev/docs | https://github.com/v8/v8"
    },
    {
      "id": "impl:spidermonkey_engine",
      "target_language": "lang:javascript",
      "implementation_name": "SpiderMonkey JavaScript/WebAssembly engine",
      "implementation_language": "C++, Rust, and JavaScript",
      "time_period": "1995-present",
      "status": "active",
      "evidence_source": "https://spidermonkey.dev/ | https://firefox-source-
docs.mozilla.org/js/index.html"
    },
    {
      "id": "impl:javascriptcore_engine",
      "target_language": "lang:javascript",
      "implementation_name": "JavaScriptCore engine (WebKit)",
      "implementation_language": "C++ with bits of C and Assembly; some
Objective-C integration",
      "time_period": "2001-present",
      "status": "active",
      "evidence_source": "https://docs.webkit.org/Getting%20Started/
Introduction.html | https://docs.webkit.org/Deep%20Dive/JSC/JavaScriptCore.html"
    },
    {
      "id": "impl:dotnet_runtime",
      "target_language": "tool:dotnet_runtime",
      "implementation_name": ".NET Runtime (runtime + libraries + host)",
      "implementation_language": "C# with significant C++ and C and some
Assembly",
      "time_period": "2014-present",
      "status": "active",
      "evidence_source": "https://github.com/dotnet/runtime | https://
devblogs.microsoft.com/dotnet/coreclr-is-now-open-source/"
    },
    {
      "id": "impl:roslyn_compilers",
      "target_language": "tool:roslyn",
      "implementation_name": "Roslyn (C# and Visual Basic compilers)",
      "implementation_language": "C# and Visual Basic .NET",
      "time_period": "2014-present",
      "status": "active",
      "evidence_source": "https://github.com/dotnet/roslyn"
    },
    {
```

```
      "id": "impl:kotlin_compiler",
      "target_language": "lang:kotlin",
      "implementation_name": "Kotlin compiler (JetBrains)",
      "implementation_language": "Kotlin with Java and native components (C/C++)",
      "time_period": "2010-present",
      "status": "active",
      "evidence_source": "https://github.com/JetBrains/kotlin | https://blog.jetbrains.com/kotlin/2016/02/kotlin-1-0-released-pragmatic-language-for-jvm-and-android/"
    },
    {
      "id": "impl:kotlin_k2_frontend",
      "target_language": "lang:kotlin",
      "implementation_name": "K2 compiler frontend (rewrite)",
      "implementation_language": "Kotlin",
      "time_period": "2021-present",
      "status": "active",
      "evidence_source": "https://blog.jetbrains.com/kotlin/2023/02/k2-kotlin-2-0/"
    },
    {
      "id": "impl:swift_compiler",
      "target_language": "lang:swift",
      "implementation_name": "Swift compiler and standard libraries (swiftlang/swift repo)",
      "implementation_language": "C++ and Swift (plus C and build tooling)",
      "time_period": "2015-present",
      "status": "active",
      "evidence_source": "https://github.com/swiftlang/swift | https://www.apple.com/newsroom/2015/12/03Apple-Releases-Swift-as-Open-Source/"
    },
    {
      "id": "impl:swift_bootstrap_swiftcompilersources",
      "target_language": "lang:swift",
      "implementation_name": "SwiftCompilerSources (Swift-in-Swift components) bootstrap flow",
      "implementation_language": "Swift (built in multiple levels using a level-0 compiler)",
      "time_period": "2022-present",
      "status": "active",
      "evidence_source": "https://github.com/apple/swift/blob/main/SwiftCompilerSources/README.md"
    }
  ],
  "edges": [
    {
      "from_language": "lang:b",
```

```
      "to_language": "lang:c",
      "relationship": "compiler_written_in",
      "start_year": 1972,
      "end_year": 1973,
      "confidence": 0.75,
      "evidence_source": "https://www.bell-labs.com/usr/dmr/www/chist.html"
    },
    {
      "from_language": "lang:cxx",
      "to_language": "lang:cxx",
      "relationship": "compiler_written_in",
      "start_year": 1983,
      "end_year": null,
      "confidence": 0.8,
      "evidence_source": "https://www.stroustrup.com/hopl2.pdf"
    },
    {
      "from_language": "lang:cxx",
      "to_language": "tool:llvm",
      "relationship": "compiler_written_in",
      "start_year": 2003,
      "end_year": null,
      "confidence": 0.95,
      "evidence_source": "https://llvm.org/docs/GettingStarted.html | https://
github.com/llvm/llvm-project"
    },
    {
      "from_language": "lang:cxx",
      "to_language": "tool:clang",
      "relationship": "compiler_written_in",
      "start_year": 2007,
      "end_year": null,
      "confidence": 0.9,
      "evidence_source": "https://discourse.llvm.org/t/new-llvm-c-front-end-
clang/55248"
    },
    {
      "from_language": "lang:cxx",
      "to_language": "tool:gcc",
      "relationship": "compiler_written_in",
      "start_year": 1987,
      "end_year": null,
      "confidence": 0.9,
      "evidence_source": "https://github.com/gcc-mirror/gcc"
    },
    {
      "from_language": "lang:c",
      "to_language": "tool:gcc",
```

```
    "relationship": "compiler_written_in",
    "start_year": 1987,
    "end_year": null,
    "confidence": 0.9,
    "evidence_source": "https://github.com/gcc-mirror/gcc"
  },
  {
    "from_language": "lang:c",
    "to_language": "lang:go",
    "relationship": "compiler_written_in",
    "start_year": 2009,
    "end_year": 2014,
    "confidence": 0.9,
    "evidence_source": "https://go.dev/doc/faq"
  },
  {
    "from_language": "lang:go",
    "to_language": "lang:go",
    "relationship": "compiler_written_in",
    "start_year": 2015,
    "end_year": null,
    "confidence": 0.95,
    "evidence_source": "https://go.dev/doc/go1.5 | https://go.dev/blog/go1.5"
  },
  {
    "from_language": "lang:assembly",
    "to_language": "lang:go",
    "relationship": "runtime_written_in",
    "start_year": 2015,
    "end_year": null,
    "confidence": 0.95,
    "evidence_source": "https://go.dev/doc/go1.5"
  },
  {
    "from_language": "lang:go",
    "to_language": "lang:go",
    "relationship": "runtime_written_in",
    "start_year": 2015,
    "end_year": null,
    "confidence": 0.95,
    "evidence_source": "https://go.dev/doc/go1.5"
  },
  {
    "from_language": "lang:ocaml",
    "to_language": "lang:rust",
    "relationship": "compiler_written_in",
    "start_year": 2006,
    "end_year": 2009,
```

```
      "confidence": 0.9,
      "evidence_source": "https://github.com/graydon/rust-prehistory"
    },
    {
      "from_language": "lang:c",
      "to_language": "lang:rust",
      "relationship": "compiler_written_in",
      "start_year": 2006,
      "end_year": 2009,
      "confidence": 0.9,
      "evidence_source": "https://github.com/graydon/rust-prehistory"
    },
    {
      "from_language": "lang:rust",
      "to_language": "lang:rust",
      "relationship": "compiler_written_in",
      "start_year": 2015,
      "end_year": null,
      "confidence": 0.9,
      "evidence_source": "https://github.com/rust-lang/rust"
    },
    {
      "from_language": "lang:machine_code",
      "to_language": "lang:rust",
      "relationship": "bootstrap_written_in",
      "start_year": 2015,
      "end_year": null,
      "confidence": 0.85,
      "evidence_source": "https://rustc-dev-guide.rust-lang.org/building/
bootstrapping/what-bootstrapping-does.html"
    },
    {
      "from_language": "lang:cxx",
      "to_language": "tool:mrustc",
      "relationship": "compiler_written_in",
      "start_year": 2016,
      "end_year": null,
      "confidence": 0.95,
      "evidence_source": "https://github.com/thepowersgang/mrustc"
    },
    {
      "from_language": "lang:c",
      "to_language": "lang:python",
      "relationship": "runtime_written_in",
      "start_year": 1991,
      "end_year": null,
      "confidence": 0.95,
      "evidence_source": "https://github.com/python/cpython | https://
```

```
en.wikipedia.org/wiki/History_of_Python"
    },
    {
      "from_language": "lang:python",
      "to_language": "lang:python",
      "relationship": "runtime_written_in",
      "start_year": 1991,
      "end_year": null,
      "confidence": 0.85,
      "evidence_source": "https://github.com/python/cpython"
    },
    {
      "from_language": "lang:c",
      "to_language": "lang:ruby",
      "relationship": "runtime_written_in",
      "start_year": 1995,
      "end_year": null,
      "confidence": 0.9,
      "evidence_source": "https://github.com/ruby/ruby"
    },
    {
      "from_language": "lang:ruby",
      "to_language": "lang:ruby",
      "relationship": "runtime_written_in",
      "start_year": 1995,
      "end_year": null,
      "confidence": 0.85,
      "evidence_source": "https://github.com/ruby/ruby"
    },
    {
      "from_language": "lang:c",
      "to_language": "lang:ocaml",
      "relationship": "runtime_written_in",
      "start_year": 1996,
      "end_year": null,
      "confidence": 0.95,
      "evidence_source": "https://github.com/ocaml/ocaml | https://ocaml.org/
docs/using-the-ocaml-compiler-toolchain"
    },
    {
      "from_language": "lang:ocaml",
      "to_language": "lang:ocaml",
      "relationship": "compiler_written_in",
      "start_year": 1996,
      "end_year": null,
      "confidence": 0.95,
      "evidence_source": "https://github.com/ocaml/ocaml"
    },
```

```
    {
      "from_language": "lang:assembly",
      "to_language": "lang:ocaml",
      "relationship": "runtime_written_in",
      "start_year": 1996,
      "end_year": null,
      "confidence": 0.85,
      "evidence_source": "https://github.com/ocaml/ocaml"
    },
    {
      "from_language": "lang:haskell",
      "to_language": "lang:haskell",
      "relationship": "compiler_written_in",
      "start_year": 1992,
      "end_year": null,
      "confidence": 0.7,
      "evidence_source": "https://en.wikipedia.org/wiki/
Glasgow_Haskell_Compiler"
    },
    {
      "from_language": "lang:c",
      "to_language": "lang:haskell",
      "relationship": "runtime_written_in",
      "start_year": 1992,
      "end_year": null,
      "confidence": 0.7,
      "evidence_source": "https://en.wikipedia.org/wiki/
Glasgow_Haskell_Compiler"
    },
    {
      "from_language": "lang:java",
      "to_language": "lang:java",
      "relationship": "compiler_written_in",
      "start_year": 2006,
      "end_year": null,
      "confidence": 0.9,
      "evidence_source": "https://github.com/openjdk/jdk/blob/master/src/
jdk.compiler/share/classes/com/sun/tools/javac/Main.java"
    },
    {
      "from_language": "lang:cxx",
      "to_language": "lang:java",
      "relationship": "runtime_written_in",
      "start_year": 2006,
      "end_year": null,
      "confidence": 0.95,
      "evidence_source": "https://openjdk.org/jeps/8283227"
    },
```

```
    {
      "from_language": "lang:cxx",
      "to_language": "tool:hotspot",
      "relationship": "runtime_written_in",
      "start_year": 2006,
      "end_year": null,
      "confidence": 0.95,
      "evidence_source": "https://openjdk.org/jeps/8283227"
    },
    {
      "from_language": "lang:cxx",
      "to_language": "tool:v8",
      "relationship": "runtime_written_in",
      "start_year": 2008,
      "end_year": null,
      "confidence": 0.95,
      "evidence_source": "https://v8.dev/docs | https://github.com/v8/v8"
    },
    {
      "from_language": "lang:cxx",
      "to_language": "tool:spidermonkey",
      "relationship": "runtime_written_in",
      "start_year": 1995,
      "end_year": null,
      "confidence": 0.9,
      "evidence_source": "https://spidermonkey.dev/ | https://firefox-source-
docs.mozilla.org/js/index.html"
    },
    {
      "from_language": "lang:rust",
      "to_language": "tool:spidermonkey",
      "relationship": "runtime_written_in",
      "start_year": 2015,
      "end_year": null,
      "confidence": 0.85,
      "evidence_source": "https://spidermonkey.dev/"
    },
    {
      "from_language": "lang:javascript",
      "to_language": "tool:spidermonkey",
      "relationship": "runtime_written_in",
      "start_year": 1995,
      "end_year": null,
      "confidence": 0.85,
      "evidence_source": "https://spidermonkey.dev/"
    },
    {
      "from_language": "lang:cxx",
```

```
      "to_language": "tool:javascriptcore",
      "relationship": "runtime_written_in",
      "start_year": 2001,
      "end_year": null,
      "confidence": 0.9,
      "evidence_source": "https://docs.webkit.org/Getting%20Started/
Introduction.html"
    },
    {
      "from_language": "lang:c",
      "to_language": "tool:javascriptcore",
      "relationship": "runtime_written_in",
      "start_year": 2001,
      "end_year": null,
      "confidence": 0.85,
      "evidence_source": "https://docs.webkit.org/Getting%20Started/
Introduction.html"
    },
    {
      "from_language": "lang:assembly",
      "to_language": "tool:javascriptcore",
      "relationship": "runtime_written_in",
      "start_year": 2001,
      "end_year": null,
      "confidence": 0.85,
      "evidence_source": "https://docs.webkit.org/Getting%20Started/
Introduction.html"
    },
    {
      "from_language": "lang:csharp",
      "to_language": "tool:roslyn",
      "relationship": "compiler_written_in",
      "start_year": 2014,
      "end_year": null,
      "confidence": 0.9,
      "evidence_source": "https://github.com/dotnet/roslyn"
    },
    {
      "from_language": "lang:csharp",
      "to_language": "tool:dotnet_runtime",
      "relationship": "runtime_written_in",
      "start_year": 2014,
      "end_year": null,
      "confidence": 0.95,
      "evidence_source": "https://github.com/dotnet/runtime"
    },
    {
      "from_language": "lang:cxx",
```

```
      "to_language": "tool:dotnet_runtime",
      "relationship": "runtime_written_in",
      "start_year": 2014,
      "end_year": null,
      "confidence": 0.95,
      "evidence_source": "https://github.com/dotnet/runtime | https://
devblogs.microsoft.com/dotnet/coreclr-is-now-open-source/"
    },
    {
      "from_language": "lang:c",
      "to_language": "tool:dotnet_runtime",
      "relationship": "runtime_written_in",
      "start_year": 2014,
      "end_year": null,
      "confidence": 0.95,
      "evidence_source": "https://github.com/dotnet/runtime"
    },
    {
      "from_language": "lang:assembly",
      "to_language": "tool:dotnet_runtime",
      "relationship": "runtime_written_in",
      "start_year": 2014,
      "end_year": null,
      "confidence": 0.85,
      "evidence_source": "https://github.com/dotnet/runtime"
    },
    {
      "from_language": "lang:kotlin",
      "to_language": "lang:kotlin",
      "relationship": "compiler_written_in",
      "start_year": 2016,
      "end_year": null,
      "confidence": 0.9,
      "evidence_source": "https://github.com/JetBrains/kotlin"
    },
    {
      "from_language": "lang:java",
      "to_language": "lang:kotlin",
      "relationship": "compiler_written_in",
      "start_year": 2016,
      "end_year": null,
      "confidence": 0.9,
      "evidence_source": "https://github.com/JetBrains/kotlin"
    },
    {
      "from_language": "lang:c",
      "to_language": "lang:kotlin",
      "relationship": "runtime_written_in",
```

```
      "start_year": 2016,
      "end_year": null,
      "confidence": 0.8,
      "evidence_source": "https://github.com/JetBrains/kotlin"
    },
    {
      "from_language": "lang:cxx",
      "to_language": "lang:kotlin",
      "relationship": "runtime_written_in",
      "start_year": 2016,
      "end_year": null,
      "confidence": 0.8,
      "evidence_source": "https://github.com/JetBrains/kotlin"
    },
    {
      "from_language": "lang:cxx",
      "to_language": "lang:swift",
      "relationship": "compiler_written_in",
      "start_year": 2015,
      "end_year": null,
      "confidence": 0.95,
      "evidence_source": "https://github.com/swiftlang/swift | https://
www.apple.com/newsroom/2015/12/03Apple-Releases-Swift-as-Open-Source/"
    },
    {
      "from_language": "lang:swift",
      "to_language": "lang:swift",
      "relationship": "compiler_written_in",
      "start_year": 2015,
      "end_year": null,
      "confidence": 0.95,
      "evidence_source": "https://github.com/swiftlang/swift"
    },
    {
      "from_language": "lang:c",
      "to_language": "lang:swift",
      "relationship": "compiler_written_in",
      "start_year": 2015,
      "end_year": null,
      "confidence": 0.85,
      "evidence_source": "https://github.com/swiftlang/swift"
    },
    {
      "from_language": "lang:machine_code",
      "to_language": "lang:swift",
      "relationship": "bootstrap_written_in",
      "start_year": 2022,
      "end_year": null,
```

```
      "confidence": 0.8,
      "evidence_source": "https://github.com/apple/swift/blob/main/
SwiftCompilerSources/README.md"
    }
  ]
}
```


Validation notes

Java's earliest compiler implementation language before OpenJDK is not consistently documented in primary sources within this run. The dataset avoids asserting a full timeline for that phase and instead anchors "javac in Java" on OpenJDK source evidence, plus the 1996 Java 1.0 press release for the presence of a compiler and VM. [4]

GHC's "Haskell + C" split is widely reported, but this run did not capture a single primary upstream page that states the implementation-language composition as plainly as the repo language summaries do for several other projects. The dataset therefore uses a verifiable summary source and sets confidence below 0.8 for those edges. [5]

Ruby 1.9's "YARV becomes official" linkage is supported by widely cited secondary sources, while the official Ruby 1.9.0 release announcement does not, by itself, state "YARV" in the snippet captured here. The dataset treats that lineage as real but assigns confidence below 0.8 when it depends on secondary confirmation. [6]

Self-hosting relationships introduce self-loops in the edge list (for example Rust->Rust and Go->Go). If your plotting pipeline requires a strict DAG, filter out edges where from_language equals to_language, then treat self-hosting from the implementations table and node notes instead. [7]

Recommended sources for further verification

For Go bootstrap and the C-to-Go transition, the Go 1.5 release notes and the Go FAQ are the most direct references, and both explicitly describe the toolchain conversion and the "implemented in Go and assembler" state. [8]

For Rust, use the rustc-dev-guide pages on bootstrapping stages (stage0, stage1, stage2) plus the historical rust-prehistory archive for early implementation-language evidence. [9]

For LLVM and Clang, the LLVM "Getting Started" documentation and the LLVM/Clang announcement mail are strong primary anchors, and the llvm-project repo gives a stable code reference. [10]

For OpenJDK HotSpot and javac, use OpenJDK's JEP on source structure (which states HotSpot code is written in C++) and the OpenJDK compiler group docs and source tree for javac. [11]

For .NET, the dotnet/runtime and dotnet/roslyn repos plus the Microsoft CoreCLR open-source announcement provide a grounded view of the runtime and compiler implementation language mix. [12]

For the JavaScript engines, rely on project-owned docs for V8 and SpiderMonkey plus WebKit's documentation for JavaScriptCore's language mix. [13]

For Swift, use Apple's open-source announcement and the swiftlang/swift repo statistics, plus SwiftCompilerSources documentation for the Swift-in-Swift bootstrap flow. [14]

Trust and bootstrapping implications

The "Trusting Trust" attack shows that source review alone does not guarantee a trustworthy compiler binary, because a compromised compiler can inject backdoors into outputs (including into future compilers) without leaving evidence in the source. [15]

Reproducible builds reduce this risk by enabling independent parties to rebuild and compare artifacts bit-for-bit from the same source, environment, and instructions. This supports stronger supply-chain verification and makes unexpected binary differences detectable. [16]

Diverse Double-Compiling (DDC) targets the Trusting Trust problem more directly by rebuilding a compiler with a second, independently trusted compiler, then comparing results to detect subversion, under stated assumptions. [17]

Stage0 and "bootstrappable builds" projects push this further by shrinking the trusted binary seed needed to build a modern stack from scratch, aiming for a verifiable chain starting from minimal, inspectable components. This shifts risk from large opaque binaries toward smaller seeds that support manual or highly audited creation paths. [18]

---

[1] [7] [8] https://go.dev/doc/go1.5
https://go.dev/doc/go1.5

[2] [9] https://rustc-dev-guide.rust-lang.org/building/bootstrapping/what-bootstrapping-does.html
https://rustc-dev-guide.rust-lang.org/building/bootstrapping/what-bootstrapping-does.html

[3] https://www.stroustrup.com/hopl2.pdf
https://www.stroustrup.com/hopl2.pdf

[4] https://github.com/openjdk/jdk/blob/master/src/jdk.compiler/share/classes/com/sun/tools/javac/Main.java
https://github.com/openjdk/jdk/blob/master/src/jdk.compiler/share/classes/com/sun/tools/javac/Main.java

[5] https://en.wikipedia.org/wiki/Glasgow_Haskell_Compiler
https://en.wikipedia.org/wiki/Glasgow_Haskell_Compiler

[6] https://www.ruby-lang.org/en/news/2007/12/25/ruby-1-9-0-released/
https://www.ruby-lang.org/en/news/2007/12/25/ruby-1-9-0-released/

[10] https://scispace.com/papers/the-development-of-the-c-language-2kqjl3tjc0
https://scispace.com/papers/the-development-of-the-c-language-2kqjl3tjc0

[11] https://openjdk.org/jeps/8283227
https://openjdk.org/jeps/8283227

12 https://devblogs.microsoft.com/dotnet/coreclr-is-now-open-source/
https://devblogs.microsoft.com/dotnet/coreclr-is-now-open-source/

13 https://v8.dev/docs
https://v8.dev/docs

14 https://www.apple.com/newsroom/2015/12/03Apple-Releases-Swift-as-Open-Source/
https://www.apple.com/newsroom/2015/12/03Apple-Releases-Swift-as-Open-Source/

15 Reflections on Trusting Trust
https://www.cs.cmu.edu/~rdriley/487/papers/Thompson_1984_ReflectionsonTrustingTrust.pdf?utm_source=chatgpt.com

16 Definitions — reproducible-builds.org
https://reproducible-builds.org/docs/definition/?utm_source=chatgpt.com

17 Countering Trusting Trust through Diverse Double-Compiling
https://dwheeler.com/trusting-trust/wheelerd-trust.pdf?utm_source=chatgpt.com

18 Bootstrappable builds
https://lwn.net/Articles/841797/?utm_source=chatgpt.com