As a baseline metric we decided to pick relevant features of programming languages for developing web applications (**Immutable objects, Exception handling, Deadlock free concurrent programming model, Race condition free concurrent programming model, Verification: invariants, pre- & postconditions**, etc) and for developing applications with complex business logic (**Unified type system, Exception handling, Object-oriented programming support in general, Inheritance model, Classes/modules as runtime objects, Concurrent programming paradigm, Support of generic entities**, etc).

Here by web applications we mean a typical client-server program, for example a simple retail store, and by a complex business logic application we understand, for example, a modern CMS system supporting hundreds of entity types with different inheritance and architectural patterns.

Keep in mind that the provided examples are just our first assumptions and the user of the tool can specify any number of features and provide a relevant metric function to create a new metric for comparison.

Here we present the scores for our 2 baseline metrics against an “ideal” PL model:

| Web Application scores: | Complex Business Application scores: |
|---|---|
| Ideal - 24 (100%) | Ideal - 54 (100%) |
| Dart - 19 (79%) | C++ - 53 (98%) |
| Swift - 18 (75%) | Kotlin - 48 (89%) |
| Kotlin - 18 (75%) | Swift - 46 (85%) |
| Rust - 18 (75%) | Rust - 45 (83%) |
| Java - 17 (71%) | Dart - 43 (80%) |
| C++ - 16 (67%) | Java - 41 (76%) |
| Go - 14 (58%) | Go - 21 (39%) |
| JavaScript - 9 (37%) | JavaScript - 18 (33%) |
| C - 6 (25%) | C - 9 (17%) |

We got mixed results, some are unexpected though:

- JavaScript scored very low for the web metric, which goes into conflict with our objective reality.
- C scored quite low for business applications mostly due to a lack of OOP and Generics, exception handling