Assignment: **Programming Languages I Might Like To Learn**

Written by **David Hennigan**

Abstract

This assignment includes 6 short texts about programming languages that peaked my interest, highlighting some of their key selling points. Many of the languages I've chosen have applications to web development, since the field interests me. The texts are ordered chronologically by the dates in which the languages originated.

Language 1: **PHP**

PHP was created mainly for the purpose of web development originating back in 1993 and not being released to the public until 1995. PHP was designed by a Danish-Canadian named Rasmus Lerdorf who was using C for web development prior to its creation. PHP was influenced largely by Perl, HTML, and C. Much of the original syntax looked incredibly similar to HTML, later being modified to be more suitable for being a programming language. ((Wikipedia PHP))

Some reasons to consider learning PHP are the following:

1. PHP features functional and object-oriented programming allowing for a wide-variety of uses and implementations.

2. PHP features a match expression which is similar to a switch statement in many programming languages but offers more by allowing itself to be assigned to a variable or returned from a function.

3. PHP has support for enumerations allowing for the ability to enforce states in the programming environment.

Language 2: **C#**

C# first appeared back in 2000 and was designed by Anders Hejlsberg while working at Microsoft, he is currently a lead architect for C# at Microsoft. C# has been influenced by many programming languages including C++, Haskell, and Java. C# has also had a large influence on many programming languages including Dart, Swift, and even Java. When C# first came out many called it a cheap imitation of Java, even the inventor of Java said it was essentially Java without security and reliability (Dietrich). I find it amazing that one of the languages many called it an imitation of it was able to also inspire. ((Wikipedia C Sharp))

Some reasons to consider learning C# are the following:

1. C# has key features including strong type checking, bounds checking, and automated garbage collection allowing for a longer software longevity and higher programmer productivity. C# could prove useful in large-scale projects with continuous development.

2. C# has polymorphism in the form of classes having the ability to implement many interfaces. This also helps allow for the success of large-scale projects. Polymorphism allows for further abstraction making large collaborative projects more viable.

3. C# provides a way to query a large variety of data sources via language integrated query. This can be incredibly useful for when dealing with SQL databases or XML documents.

Language 3: **Scala**

Scala first appeared on January 20th, 2004 in Switzerland and was designed by Martin Odersky. Odersky worked on Generic Java and javac before creating Scala, in fact Scala was designed to address criticisms of Java. Scala has been influenced by Common Lisp, Eiffel, Erlang, F#, Haskell, Java, Ocaml, and Smalltalk. Scala also has influenced some languages like F#, C#, and Kotlin. ((Wikipedia Scala))

Some reasons to consider learning Scala are the following:

1. Scala supports both object-oriented programming and functional programming allowing for a variety of ways to tackle problems making it a general-purpose programming language. Scala can be used for a variety of application domains.

2. Scala can be compiled to Java bytecode so Java libraries can be utilized. Scala can also be compiled to JavaScript to run in a browser allowing for use of JavaScript libraries as well. Since Scala has access to its own libraries, Java's libraries, and JavaScript's libraries a programmer can easily carry their tool belt from other languages to use in Scala.

3. Scala has many functional programming features including currying, immutability, lazy evaluation, and pattern matching. All of the previously listed features make Scala useful for applications of functional programming and improving functional programming skills.

Language 4: **Clojure**

Clojure first appeared in 2007 and was designed by Rich Hickey. Clojure took inspiration from a large array of languages including C#, C++, Java, Common Lisp, Haskell, and more. Clojure is a modern dialect of Lisp and runs on the Java platform. Clojure has given some inspiration to Elixer, Hym LFE, Pixie, and Rhine. ((Wikipedia Clojure))

Some reasons to consider learning Clojure are the following:

1. Clojure is a modern Lisp designed for concurrency, if using a lisp and concurrency are a priority then Clojure is a must have in your tool belt!

2. As mentioned earlier Clojure runs on the Java platform allowing full support of calling Java code from Clojure. Tools like Maven used in Java can also be utilized by Clojure.

3. Clojure's Lisp macro system is very similar to the one used by Common Lisp making it easier to pick up if one has past experience with Common Lisp.

Language 5: **Go**

Go was designed back in 2007, not being publicly announced until 2009, at Google by Robert Griesemer, Rob Pike, and Ken Thompson. Go is syntactically similar to C but has key features C lacks like memory safety and garbage collection. A fun fact about Go is it is often referred to as Golang because of a prior domain name. Go was largely influenced by C with the intention to add features Google felt other languages they were using at the time lacked. ((Wikipedia Go))

Some reasons to consider learning Go are the following:

1. Go has the run-time efficiency of c without having to worry about problematic features like manual memory allocation and freeing. Go proves to be a viable alternative for c when a program needs to be efficient.

2. Go has generics allowing for more code re-usability and scalability, allowing Go to be useful for large projects.

3. Go has concurrency and can support parallel programming. Allowing the language to be insanely useful for certain data structures and algorithms including divide and conquer ones.

Language 6: **Dart**

Dart first came to light in 2011 as a result of efforts from the designers Lars Bak and Kasper Lund. Dart was also developed by Google, similarly to Go, but this time with the intention of dealing with client development such as for mobile applications or websites. Dart was largely influenced by C and JavaScript, originally Dart was designed to work in the browsers using darts own virtual machine but this implementation was scrapped after Google received some backlash. After all the criticism Dart was redesigned to compile to JavaScript. ((Wikipedia Dart))

Some reasons to consider learning Dart are the following:

1. Dart as with Go features a C-style syntax making it easier to pickup after prior experience with either Go or C.

2. Dart code in some cases can run faster than the equivalent JavaScript code. Allowing for websites utilizing the language to have faster response times.

3. Dart is packaged with a standard library that provides all the necessary features to design a modern app or custom web server. This can be useful for saving time and ensuring a team has all the necessary libraries.

Citations

Dietrich, E. (2022, July 13). *C# version history: Examining the language past and present*. NDepend. Retrieved January 28, 2023, from https://blog.ndepend.com/c-versions-look-language-history/

Wikimedia Foundation. (2014, May 8). *Clojure (programming language)*. Wikipedia. Retrieved January 26, 2023, from https://en.wikipedia.org/wiki/Clojure_(programming_language)

Wikimedia Foundation. (2023, January 13). *Dart (programming language)*. Wikipedia. Retrieved January 26, 2023, from https://en.wikipedia.org/wiki/Dart_(programming_language)

Wikimedia Foundation. (2023, January 21). *C sharp (programming language)*. Wikipedia. Retrieved January 26, 2023, from https://en.wikipedia.org/wiki/C_Sharp_(programming_language)

Wikimedia Foundation. (2023, January 21). *Scala (programming language)*. Wikipedia. Retrieved January 26, 2023, from https://en.wikipedia.org/wiki/Scala_(programming_language)

Wikimedia Foundation. (2023, January 25). *PHP*. Wikipedia. Retrieved January 26, 2023, from https://en.wikipedia.org/wiki/PHP

Wikimedia Foundation. (2023, January 26). *Go (programming language)*. Wikipedia. Retrieved January 26, 2023, from https://en.wikipedia.org/wiki/Go_(programming_language)