SDV602 COmparison

Bob Win-Donnelly - 12785549

Table of Contents

[1. Heritage and Philosophy 2](#_Toc87372736)

1.1 JavaScript2

1.2 Python2

2. Development Tools 3

2.1 JavaScript3

2.2 Python3

3. How Code is Run 4

3.1 JavaScript4

3.1 Python4

4. Comparison 5

[5. References 6](#_Toc87372739)

# Heritage and Philosophy

The history or heritage of a programming language, why it was developed or its philosophy. The problem it was designed to solve and who or what continued to shape it after the creator developed the programming language are all relevant to understanding how to use that programming language today.

1. JavaScript

In 1995 a Netscape developer Brendan Eich developed a scripting language in ten days which was the beginnings of JavaScript then called Mocha (‘The History of JavaScript’, 2019). This was to be able to compete with Microsoft’s Internet Explorer browser which had 94% market share.

In the beginning JavaScript was developed for client-side interactivity, because of the range of UI elements and functionality needed to be supported before design standardisation. JavaScript had to have the flexibility to be used across many designs of UI around the web.

In 1997 because of the growth of JavaScript Netscape handed the maintenance of JavaScript to the ECMA (European Computer Manufacturers Association) which makes specifications for ECMAScript which JavaScript follows. ECMA is still updating specifications today (*JavaScript Language Resources - JavaScript | MDN*, n.d.).

The JavaScript community is not about JavaScript itself but more about web development events that are held. There are pockets of JavaScript first enthusiasts e.g., JSCamp and ECMAScript conferences but apart from that there are only small-scale local JS communities. If you counted communities for JavaScript frameworks there are many communities fractured across development tools like React and Node (*Node.Js Community*, n.d.).

1. Python

Python was released as an open-source programming language in 1991 by Guido van Rossum (*Guido van Rossum*, n.d.). While he had the idea to create Python in 1989 while working at CWI working on an Amoeba microkernel distributed system, he did this on his own time because he wanted a language that was easy to read/develop in instead of working with C.

Python was developed and maintained with four guiding principles behind every feature decision and was clarified to the wider Python community in 1999.

It should be an easy and intuitive language, just as powerful as major competitors.  
It should be open source, so anyone can contribute to its development.  
Its code should be understandable as plain English.  
It should be suitable for everyday tasks, allowing for short development times.

The quality control for these four principles is the PEP (Python Enhancement Proposal) system where developers submit a proposal for an enhancement for python with the code and documentation for how it will work (*PEP 0 -- Index of Python Enhancement Proposals (PEPs)*, n.d.).

Until 2018 Guido van Rossum was the BDFL (Benevolent Dictator for Life) who accepted PEPs but after a controversial PEP was accepted, he resigned as BDFL then the Python governing structure moved to a steering council where developers are elected to steer the development of Python instead of the BDFL structure.

Python has a large community that refer to themselves as Pythonistas and regularly hold meetings and take part in the steering council and large organisations like PyCon conferences. There are also tight knit international communities like PyLadies providing community at every level of Python (*PyCon US*, n.d.).

# Development Tools

Every language has tools that have been programmed by developers to solve a certain issue. Instead of having to build something out of nothing every time code libraries can be built upon instead. The other tool is an IDE (Integrated Development Environment) which can package those tools for you to use while having an area to input code/text.

1. JavaScript

JavaScript is spoilt for choice in the IDE department. JetBrains has built WebStorm in 2010 (*10 Best JavaScript IDE & Source Code Editors to Use [Updated]*, n.d.) as a one stop shop for JavaScript development. Visual Studio Code has constant development for all top languages including JavaScript.

In 2009 NPM (Node Package Manager) was developed by Isaac Z. Schluter is a package manager for Node.js which is server-side JavaScript developed in 2009. It is an easy-to-use package manager which can install dependencies system wide or only within the project, so it does not create conflicts with the development environment between projects (*The Secret History behind the Success of Npm and Node - Coffee Talk: Java, News, Stories and Opinions*, n.d.).

The userbase of JavaScript has developed many libraries related to web development from the traditional client side to server side and modelling. Although you can find packages for data modelling and visualisation for JavaScript, they are outnumbered by web development packages (‘The History of JavaScript’, 2019).

1. Python

Python has more IDEs specifically made for Python. Those being PyCharm, Thonny.

PyCharm even has a stripped-down IDE which is PyCharm Edu which can be configured to run tests on the solution the student has used on the problem the tutor set. The student can keep trying until the tests pass then the tutor can mark their solution (Python, n.d.-a).

Thonny is another IDE for students or beginners to Python, but it is also a very basic IDE and is not meant for professional developers.

Visual Studio Code fully supports Python as it is a major language and as an open-source code editor. The more people that use a language the more people making plugins for Visual Studio Code to ensure Python is supported.

Python3 comes with a virtual environment creation tool called venv. It works by having a semi contained environment which when run or installed to does not update the base python install. If you did not have different environments for each module/dependency, then it would be nearly impossible to work on multiple projects on one machine (Python, n.d.-b).

PIP is a recursive acronym for “pip installs python” or “pip installs packages” and is the package installer that is bundled with Python. PIP is the package manager most Pythonistas use (*Installing Python Modules — Python 3.10.0 Documentation*, n.d.).

The libraries that are installed the most with PIP are data science i.e., data manipulation, conversion, visualisation, and representation. There are web development python modules but by far what libraries are used are for data science (Python, n.d.-c).

# How Code is Run

Programming languages use interpreters or compilers to take human readable code and interpret or compile it into machine instructions which is hard to impossible for humans to read. Interpreters run code line by line at runtime and do not need to be compiled, Compiled languages compile before the code is run (*Interpreter*, n.d.). Languages can be dynamically typed which means they do not declare types in code and check the types at runtime or statically typed they declare types in code and check the types at compile time (*Typing*, n.d.). This leads us to say that interpreted languages are dynamically typed and static typed languages are compiled languages. These are broad categories and can overlap.

Programming languages can be strongly or weakly typed. Strongly typed languages will throw an error if the typing in the code does not match. Weakly typed languages will implicitly convert one type to another if it makes sense to the interpreter to do so. This may not make sense for a human before you learn how it does it (*Typing*, n.d.).

They also have objects, mutable and immutable data types. Mutable datatypes mean that it can be changed. Immutable datatypes mean they cannot be changed (Koenig-Bautista, 2019). Object oriented languages can be class based or prototype-based object-oriented languages (*Class-Based vs. Prototype-Based Languages*, n.d.). There are other types of languages for example C is not object orientated based (*Procedural - Why Is C Not Considered an ‘object-Oriented’ Language?*, n.d.).

* 1. JavaScript

JavaScript is an interpreted, dynamically, weak typed, high programming language. Which means it is run line by line instead of compiled, does not declare types in code and makes implicit conversions between types at runtime to make the code work, sometimes breaking your applications outcomes without the developer knowing.

JavaScript is a prototype object orientated language which means that any constructor or function can act as an object constructor. Especially since functions are objects. Instead of inheriting a class the object has a copy made with a new memory reference then that copy is manipulated not the original object. Constructors can be used but they are unnecessary.

Everything in JavaScript is an object, except primitive datatypes which are immutable (*Primitive - MDN Web Docs Glossary: Definitions of Web-Related Terms | MDN*, n.d.).

* 1. Python

Python is an interpreted, dynamically, strong typed programming language. The main point is that Python does not implicitly convert types at run time, it throws an error instead of trying to reconcile two types that should not be changing each other. Code is run line by line and does not declare types in code.

Python is a class-based object orientated language which means that the class is inherited/instantiated from the main class instead of it just being prototyped from an existing object.

Everything in Python is an object. But objects can be immutable by Pythons’ definition. Such as immutable list where JavaScript arrays can never be immutable.

# Comparison

In terms of the history and philosophy of JavaScript/Python we see that Python had a more structured approach over two years to building Python to be an easy-to-use language while working on microservices. JavaScript was made over ten days to solve a UI problem Netscape had competing with Microsoft’s Internet Explorer but was flexible enough to be changed with the needs of future developers.

They both had continual development by governing structures, but Python was built by the Python community PEP system while the ECMA developed specifications of how JavaScript will be used and developed. This shows in that the JavaScript community defines itself as the web development community or splinter groups forming around JavaScript frameworks like Node. While Python members call themselves Pythonistas and are driven to the community by the Python language itself.

This history defines the languages’ philosophy to this day as shown by the way the languages gravitate towards different work such as web development or data science even though both languages can easily be used to do data science or web development.

The development tools Python and JavaScript have available are immense because both are popular languages, they have wide support from IDEs with JavaScript and Python specific IDEs on offer as well as wide support from general use IDEs like Visual Studio Code. They are easy to set up and run on any machine.

Python and JavaScript have many packages to help with any part of development as I mentioned in the history section of this comparison. The main difference is the abundance of what type of libraries are available. The best thing Python has that I wish JavaScript did would be Python Edu plugin for PyCharm. It is amazing for students to learn Python from their teachers.

How the code is run differs greatly with JavaScript and Python it is along with the history and philosophy of the two languages the greatest divergence from each other. While the development tools are about equal the way the code is interpreted and typed is far different.

While they are both interpreted languages the way their interpreters run is vastly different. Pythons’ interpreter is installed to the local machine, and you can install different interpreters for Python. The JavaScript interpreter runs in the browser and there are different versions for different browsers. For Chromium based browsers it is Google’s V8 engine and for Mozilla Foundations’ Firefox which is founded by the JavaScript creator Brendan Eich it is Spidermonkey. Spidermonkey was also created by Brendan Eich at Netscape.

They are both dynamically typed as a point of similarity from two disparate languages as is the fact they are interpreted. It means you do not have to define a type in the code to be checked at compile time the type is checked as the code is run by the interpreter.

Now even though they are interpreted and dynamically typed their typing is still different as JavaScript is weak and Python is strong. That means that Python has strong typing that does not allow code to finish running if a type is mismatched i.e., number to a string conversion that is not explicitly stated in the code will cause the code to fail. While in JavaScript the weak typing means that if you did not use the strict equality operator then it will automatically convert the number to a string which will probably make any value returned from that function worthless to your purposes, but it will still run and confuse you.

In effect even though both languages are interpreted, wildly popular and are around the same age they are different while still being able to do a lot of the same tasks.

# References

*10 Best JavaScript IDE & Source Code Editors to Use [Updated]*. (n.d.). Hackr.Io. Retrieved 15 November 2021, from https://hackr.io/blog/best-javascript-ide-source-code-editors

*Class-based vs. Prototype-based languages*. (n.d.). Li60. Retrieved 18 November 2021, from https://li60.zendesk.com/hc/en-us/articles/115002448751-Class-based-vs-prototype-based-languages

*Guido van Rossum*. (n.d.). CHM. Retrieved 26 July 2021, from https://computerhistory.org/profile/guido-van-rossum/

*Installing Python Modules—Python 3.10.0 documentation*. (n.d.). Retrieved 16 November 2021, from https://docs.python.org/3/installing/index.html

*Interpreter*. (n.d.). Retrieved 18 November 2021, from https://cs.stanford.edu/people/nick/py/python-interpreter.html

*JavaScript language resources—JavaScript | MDN*. (n.d.). Retrieved 15 November 2021, from https://developer.mozilla.org/en-US/docs/Web/JavaScript/Language\_Resources

Koenig-Bautista, A. (2019, December 12). *Everything in JavaScript Is (Not) an Object*. Medium. https://betterprogramming.pub/everything-in-javascript-is-an-object-except-for-when-it-isnt-305bc65a3410

*PEP 0—Index of Python Enhancement Proposals (PEPs)*. (n.d.). Python.Org. Retrieved 15 November 2021, from https://www.python.org/dev/peps/

*Primitive—MDN Web Docs Glossary: Definitions of Web-related terms | MDN*. (n.d.). Retrieved 23 March 2021, from https://developer.mozilla.org/en-US/docs/Glossary/Primitive

*procedural—Why is C not considered an ‘object-oriented’ language?* (n.d.). Software Engineering Stack Exchange. Retrieved 18 November 2021, from https://softwareengineering.stackexchange.com/questions/113533/why-is-c-not-considered-an-object-oriented-language

*PyCon US*. (n.d.). Retrieved 18 November 2021, from https://us.pycon.org/2020/schedule/presentation/135/

Python, R. (n.d.-a). *Python IDEs and Code Editors (Guide) – Real Python*. Retrieved 16 November 2021, from https://realpython.com/python-ides-code-editors-guide/

Python, R. (n.d.-b). *Python Virtual Environments: A Primer – Real Python*. Retrieved 16 November 2021, from https://realpython.com/python-virtual-environments-a-primer/

Python, R. (n.d.-c). *Python vs JavaScript for Pythonistas – Real Python*. Retrieved 15 November 2021, from https://realpython.com/python-vs-javascript/

The History of JavaScript: Everything You Need to Know. (2019, August 19). *Springboard Blog*. https://www.springboard.com/blog/data-science/history-of-javascript/

*The secret history behind the success of npm and Node—Coffee Talk: Java, News, Stories and Opinions*. (n.d.). Retrieved 15 November 2021, from https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/The-secret-history-behind-the-success-of-npm-and-Node

*Typing: Dynamic vs. static and weak vs strong | Intro to Programming*. (n.d.). Retrieved 18 November 2021, from https://en.hexlet.io/courses/intro\_to\_programming/lessons/types/theory\_unit