# ArchiveXojo and Git

It is important to understand that **ArchiveXojo** and **Git** are not mutually exclusive and are not incompatible with each other. In some areas, their functionality overlaps. In other areas, they each provide services that the other does not. It is entirely possible to use **ArchiveXojo** and **Git** simultaneously and, indeed, I often do. Use of one does not preclude use of the other.

## Simple Git

**Git** is a very powerful tool with a large range of functionality. Entire books are devoted to explaining the many facets of **Git**. It has a large spectrum of complexity. It can be used by a solo developer all on a local computer or by multiple developers scattered around the planet collaborating on a project with the **Git** files hosted on an external server accessible to those developers.

I am not a **Git** expert. Far from it. But it is important to realize that **Git** can be used by a Xojo developer at a simple level without grokking the entirety of **Git**. For someone unfamiliar with **Git** and considering its use, understand that the cost of entry is low. Literally, in terms of dollars, because the basic tools are free. But also in terms of intellectual investment, because it is possible to start using **Git** understanding only a small fraction of it.

For a Xojo user interested in **Git**, but new to it, I would refer to the two part Xojo webinar given by Justin Elliot - *Getting to Know Git*. That webinar provides a solo Xojo developer with enough information to get started. It explains how to use **Git** with *SourceTree*, a tool which offers a graphical user interface.

In the most basic terms, **Git** allows you to designate a folder as one that **Git** "watches". At intervals, determined by the user, **Git** takes a snapshot of this folder. It records, in a very efficient fashion, all the changes in that folder that have occurred since the last snapshot. And it remembers all these different snapshots in such a way that the user can easily compare the state of the folder's contents at the time of one snapshot with its state at the time of another. It is possible to go back in time, resurrecting the state of the folder as it existed at the time of an earlier snapshot.

Developing software is an endeavor in which one finds oneself occasionally entering some tunnel and only to realize some time later that it is a deadend, a bad idea. **Git** provides tools to extract yourself.

Xojo, by default, saves your work as a binary file. **Git** is capable of comparing the state of two binary files, but this is not optimal for the user trying to understand the differences. As explained in Justin's webinar, it is possible to save your Xojo projects as a multitude of text files - (Xojo Project). If these files are saved to a to a folder that **Git** is

watching, the human user can comprehend, within **Git**, the development progress as as it moves from one snapshot to the next.

## Comparing and Contrasting ArchiveXojo and Git

Above, I have tried to provide enough information about **Git** that the reader can understand how it differs from **ArchiveXojo**. Understand that complex **Git** goes far beyond what is described above. Coordinated, multi-developer project development with branching and external server hosting are all things that **Git** can provide and ArchiveXojo does not touch. But what about the basics?

**Git**, when used as outlined above, saves *everything* about the project. **ArchiveXojo** deals *only* with code. **Git** saves snapshots of the code but also all the UI elements, the window sizes, the position and name and state of controls, etc. **ArchiveXojo**, dealing only with code, is simpler but less comprehensive.

**ArchiveXojo** is intended to save all your Xojo code from *all* your Xojo projects in a single database. In contrast, **Git** deals with each project individually. **ArchiveXojo** records the changes of your code as it is developed; it tends to offer a more granular look at the code progression. **ArchiveXojo** takes snapshots of individual methods and code handlers as often as the user desires. Whenever I make a change to some method, I record it for **ArchiveXojo** before moving to some other code handler or before running and testing the project. There is no need to leave Xojo. Each one of these changes is recorded with a time stamp and can be subsequently reviewed in **ArchiveXojo**. **Git** snapshots are generally acquired less frequently and not at the level of the individual method. The entire project is saved. **Git** is launched and a snapshot obtained of all changes in all the code and UI before returning to Xojo.

**ArchiveXojo** becomes a repository of all the versions of all your code from all your projects in a single database. It offers search tools that one would expect froma classic database. It is easy, for example, to search all your code from all your projects for the use of a particular Xojo command. It is a resource for your future development in Xojo.

**ArchiveXojo** initially saves all its entries in a text file that exists independent of your Xojo development. This can provide a safety net should Xojo suffer some unusual and serious crash. At least your own recent code changes are preserved which can help in restoring your work.

| ArchiveXojo | Git |
|---|---|
| Searchable Classic Database of Code & Its Versions | Record of State of All Code & UI at Time Intervals |
| Many Snapshots of Code at Level of Individual Method with Time Stamps | Fewer Snapshots at Level of Entire Project |
| Code Only | Entire Project (Code & UI) |
| Code From Multiple Projects in One Database | Git Archive on a Project by Project Basis |
| Possibility of Recovering and Utilizing Earlier Versions of Code | Roll Back of Entire Project to Earlier Versions |
| Documentation of Code Changes on an Ongoing Basis in a Text File During Xojo Development | Exit Xojo and Enter Git Tools to Record the Current State of the Project |
| Many Developers Could Contribute Code to a Single ArchiveXojo Database | Extensive Collaborative Tools for Multiple Developers |
| … | Many Tools for Managing and Organizing Code Projects: Branching, Server Archives, etc. |

## ArchiveXojo and GIT can be Used Together!