[0401] Why Immutability is A Good Thing

Over time, programmers have learned that some of the power and flexibility that languages like C++, C#, and Java give you don't always result in better programs.

When these languages first appeared, programmers (and the language designers) went crazy adding as many new language "features" as they could think of. Many of these new features proved to be very helpful and are still in heavy use today. Other features, however, started cropping up again and again in code that was buggy and difficult to maintain.

One of the problem areas turned out to be data types that allowed programmers to change their internal values. Strings are a good example. In an effort to be more efficient, programmers used to create programs that directly modified the characters inside of a string. (C and C++ programmers still do that frequently today.) This often led to bugs, especially in multi-threaded programs.

Over time, the concept of "mutability" (i.e., change-ability) was called into question and a "Best Practice" emerged that says "**Favor immutability over mutability.**" In other words, don't let other programmers reach in and change your data values arbitrarily.

When newer OOP languages like Java and C# were introduced, their designers deliberately made it more difficult to create mutable strings. There was some grumbling at first, but now it is very common practice to just create a new string and abandon the previous version (for the Garbage Collector to free up later).

We will revisit this topic later when we get to OO Design.